Extended Essay

Computer Science

To what extent does Android Runtime improve upon the Dalvik Virtual Machine in the performance of the Android Operating System?

Candidate Number: xxx
Centre Number: xxx

Word Count: 3853

Abstract:

This essay examines the extent that Android Runtime improves upon the Dalvik Virtual Machine in performance of the Android Operating System. Performance was measured using four factors; speed, memory use, storage size and battery life, to gauge the effectiveness of each runtime. Android version 4.4 provided users with the choice between two runtimes and therefore, due to the wide distribution of this version, the performance of each is important. Initially the experiment focused on the theoretical improvements that should arise from the new runtime with the first being in relation to garbage collection. The number of pauses during the process was halved which decreased the time where the application was paused, reducing the number of dropped frames thus increasing fluidity for the user. The second improvement was the replacement of the Just-in-Time compiler by the new Ahead-of-Time compiler. This allowed applications to be compiled to native code at the point of installation which reduces the battery use, processing power and the application start-up time but also increases the installation time and storage usage. The final improvement was the introduction of kernel same-page merging which reduced duplicates of memory pages in RAM. Experiments were then completed to find out how the user was effected. These included an increase in application opening speeds, evidence of a decrease in the time spent in garbage collection and a reduction in the number of dropped frames. Furthermore, multiple benchmarks were run which highlighted an increase in performance in many different areas; however, negative effects were also found with an increase in the storage size of applications between 4% and 87% and an increase in installation times arising. This information leads to the conclusion that the new runtime was generally beneficial in terms of performance and, therefore, users with the option should choose the new runtime.

Word Count: 300

Contents

1. Introduction

The Android operating system is an open source operating system purchased and developed by Google, initially with the intention to run on mobile phones. It started out as a less popular system which was struggling to compete with the competition; however, over time its user base has grown and Android had, in early 2014, sold almost a billion devices, with 1.5 million being activated each day[1]. This goes to show its importance in the lives of many people around the world, as approximately 84.6% of global smartphone shipments housed Android as their operating system in 2014[2].

1.1 *The Research Question*

My research question examines the extent to which Android Runtime (ART) improves upon the Dalvik Virtual Machine. The extent of this will be measured through performance in terms of speed, memory use, size and battery life. This question is significant due to Android's high user base and the fact that most manufacturers and carriers alter the operating system with their own software to try and provide more features. These alterations to the operating system create difficulties for the manufacturers and carriers when they want to update the system to Google's latest release because they must integrate their old changes into the new system. Due to this and the cost required to make the changes, many phones still run older iterations of the OS and lack the performance improvements of newer software. This makes it more important to discover how to make these devices best perform.

With version 4.4 of the Android operating system, users were given a choice between ART and Dalvik. ART was an experimental runtime that was developed to try and increase the performance of the operating system over the previously

[1] Amadeo, R. (2014). *The history of Android*. [online] Ars Technica. Available at: http://arstechnica.com/gadgets/2014/06/building-Android-a-40000-word-history-of-Googles-mobile-os/ [Accessed 13 Apr. 2015].
[2] Hornyak, T. (2014). *Android grabs record 85 percent smartphone share*. [online] PCWorld. Available at: http://www.pcworld.com/article/2460020/Android-grabs-record-85-percent-smartphone-share.html [Accessed 13 Apr. 2015].

favoured Dalvik Virtual Machine by changing how applications on the device install and run[3].

The number of users on different versions of the Android operating system can be seen in figures 1.1.1 and 1.1.2 which demonstrate that the majority of devices run version 4.4. Subsequently, it is important to know which runtime this large number of users should use on their system to boost performance.

Furthermore, the next generation of the Android operating system (version 5.x) comes with Android Runtime as a default[4], making the knowledge of how the runtimes compare very significant.

Figure 1.1.1[5] - A table to show the distribution of different Android versions that accessed the Google Play Store over 7 days ending on May 4th 2015.

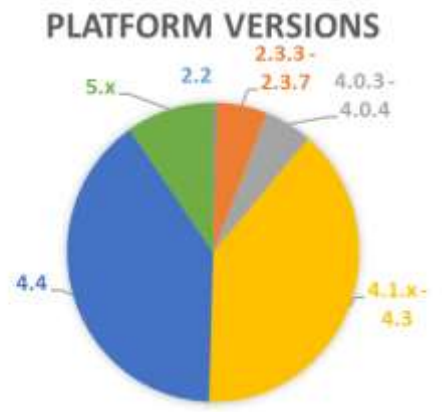
Version	Codename	Distribution (%)
2.2	*Froyo*	0.3
2.3.3 - 2.3.7	*Gingerbread*	5.7
4.0.3 - 4.0.4	*Ice Cream Sandwich*	5.3
4.1.x - 4.3	*Jelly Bean*	39.2
4.4	*KitKat*	39.9
5.x	*Lollipop*	9.7

[3] Toombs, C. (2013). *Meet ART, Part 1: The New Super-Fast Android Runtime Google Has Been Working On In Secret For Over 2 Years Debuts In KitKat*. [online] Android Police. Available at: http://www.Androidpolice.com/2013/11/06/meet-art-part-1-the-new-super-fast-Android-runtime-Google-has-been-working-on-in-secret-for-over-2-years-debuts-in-kitkat/ [Accessed 3 May 2015].

[4] Frumusanu, A. (2014). *A Closer Look at Android Runtime (ART) in Android L*. [online] Anandtech.com. Available at: http://anandtech.com/show/8231/a-closer-look-at-Android-runtime-art-in-Android-l/ [Accessed 1 May 2015].

[5] Developer.Android.com, (2015). *Dashboards | Android Developers*. [online] Available at: https://developer.Android.com/about/dashboards/index.html [Accessed 08 May 2015].

Figure 1.1.2[6] - A pie chart displaying the information displayed in figure 1.1.1 that portrays the percentage of users on different versions of the Android operating system.



1.2 *International Significance*

Android as an operating system is used on more devices than just mobile phones. An example of this is a project by a company called Keepod. Keepod is a USB drive costing $7 that has a version of Android 4.4 on it specialised for use on a desktop. These devices are being donated to people in developing countries along with old laptops which gives each person the ability to have their own computer with separate files and a personalised and safe experience.

This scheme is proof of the significance of the question, as the Keepods are used on old laptops which have slow and old processors and therefore, making the most of the processing power and battery life, is essential[7].

[6] Ibid

[7] Simmons, D. (2014). *Keepod: Can a $7 stick provide billions computer access? - BBC News*. [online] BBC News. Available at: http://www.bbc.co.uk/news/technology-27346567 [Accessed 4 May 2015].

2. Theory

To compare Android Runtime and the Dalvik Virtual Machine, it is important to know what they are used for and how they work. From this it can be seen how they differ and why the new runtime was introduced.

2.1 Virtual Machines

A virtual machine on a computer allows it to emulate a different system, these come in two main types. First is a system virtual machine where a complete operating system (for example Windows) can be loaded onto a system with virtual memory and specifically allocated system resource access (including storage, processor speed and network connectivity)[8]. This provides many possibilities for the user for example testing software where any other data on the computer can be protected[9]. The second type of virtual machine is a process virtual machine which is specifically designed to run a specific program or process. Both Android Runtime and the Dalvik Virtual Machine are examples of a process virtual machine as they are specialised machines made to convert the high-level code that is written by programmers in Java, to the native machine code that is run on each specific processor. This is especially important as they are designed to allow programs to work the same, independent of the underlying hardware[10].

2.2 The Dalvik Virtual Machine

The Dalvik Virtual Machine was originally created for the Android operating system by Dan Bornstein and his team at Google to make it possible to interpret (change into machine code which is understood by the hardware) compiled bytecode on the constrained resources of a mobile. Bytecode is a numeric instruction set which is designed to be executed efficiently by a software interpreter. It is made of numeric code, constants and references and it was created to aid the performance of

[8] Smith, J. and Nair, R. (2005). The architecture of virtual machines. *Computer*, [online] 38(5), pp.32-38. Available at: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1430629 [Accessed 17 Sep. 2015].
[9] Caprio, G. (2006). *Virtual Machines: Virtualization vs. Emulation*. [online] Griffincaprio.com. Available at: http://www.griffincaprio.com/blog/2006/08/virtual-machines-virtualization-vs-emulation.html [Accessed 17 Sep. 2015].
[10] Reilly, D. (2006). *Inside Java : The Java Virtual Machine*. [online] Javacoffeebreak.com. Available at: http://www.Javacoffeebreak.com/articles/inside_Java/insideJava-jan99.html [Accessed 17 Sep. 2015].

interpretation; it is generally unreadable by humans[11]. These constraints include a slow central processing unit, a small amount of RAM (high speed storage that holds the OS, applications and data currently in use[12]) and no SWAP space[13] (this is the ability to place unused items in memory onto the hard drive to free memory[14]). This means that it is not possible to increase the amount of RAM for a system using the secondary memory (storage which is slower than RAM and used to hold user data), something that most computer operating systems are able to do[15]. This is a negative factor because having a limited amount of RAM restricts the processes which can run together; however, secondary memory is much slower to access than RAM and therefore using it can considerably slow down a system.

The biggest constraint, however, is the fact that mobile devices are powered by batteries with a low capacity which does not seem to be improving. Battery technology doesn't appear to be following Moore's Law, which is a computing term introduced in 1965, that roughly implies processor speed will double every two years[16]. The capacity of batteries has only increased slightly (at a rate of 4% and is decreasing each year) whereas the amount of power that is required by other hardware is increasing at a faster rate[17]. This makes the need for power management in software and hardware exceptionally important.

When programming applications for Android, the application code is often written in a high-level language such as Java due to the fact that it is easier to learn and remember than machine code. This high-level code is then compiled to bytecode for use in the Java virtual machine which is done by a Java compiler. This Bytecode is

[11] Venners, B. (1996). *Bytecode basics*. [online] JavaWorld. Available at: http://www.Javaworld.com/article/2077233/core-Java/bytecode-basics.html [Accessed 17 Sep. 2015].
[12] Rouse, M. (2005). *RAM (random access memory)*. [online] SearchStorage. Available at: http://searchstorage.techtarget.com/definition/RAM-random-access-memory [Accessed 17 Sep. 2015].
[13] Bornstein, D. (2010). *Google I/O 2008 - Dalvik Virtual Machine Internals*. [online] YouTube. Available at: https://www.youtube.com/watch?v=ptjedOZEXPM [Accessed 30 Apr. 2015].
[14] Centos.org, (n.d.). *5.1. What is Swap Space?*. [online] Available at: https://www.centos.org/docs/5/html/5.2/Deployment_Guide/s1-swap-what-is.html [Accessed 17 Sep. 2015].
[15] Differencebetween.info, (n.d.). *Difference between Virtual Memory and Swap Memory*. [online] Available at: http://www.differencebetween.info/difference-between-virtual-memory-and-swap-memory [Accessed 8 May 2015].
[16] Mooreslaw.org, (n.d.). *Moore's Law*. [online] Available at: http://www.mooreslaw.org/ [Accessed 30 Apr. 2015].
[17] Epec Engineered Technologies, (2014). *New Battery Technology Developments*. [online] Slideshare.net. Available at: http://www.slideshare.net/epectec/new-battery-trends [Accessed 17 Sep. 2015].

then translated to Dalvik Bytecode, which is stored as .dex (Dalvik Executable) or .odex (Optimised Dalvik Executable) using the conversion tool dx[18]. These files can then be compiled and executed by the Dalvik Virtual Machine.

The importance of using a virtual machine to interpret compiled Java binary code, also known as bytecode, is the ability for almost any device with the virtual machine to run the instructions[19]. Due to the nature of the operating system running on a wide variety of devices and processors, it would be very difficult to write unique machine code for every device. Therefore, targeting a virtual machine makes it possible for a developer to create their application for all devices.

A Just-in-Time compiler was introduced to the virtual machine in 2010 via Android version 2.2 which worked to reduce the time spent interpreting code. This time spent was much longer for highly computational processes and therefore those applications suffered in performance. It works by discovering the highly computational processes which are then separately interpreted and compiled into native code (code specific for the devices hardware). This is then accessed and executed without the interpretation needing to be completed again[20].

[18] forensic blog, (2012). *Comparison of Dalvik and Java Bytecode*. [online] Available at: http://forensics.spreitzenbarth.de/2012/08/27/comparison-of-dalvik-and-Java-bytecode/ [Accessed 1 May 2015].
[19] Rouse, M. (2006). *What is Java virtual machine (JVM)? - Definition from WhatIs.com*. [online] SearchSOA. Available at: http://searchsoa.techtarget.com/definition/Java-virtual-machine [Accessed 30 Apr. 2015].
[20] Cheng, B. and Buzbee, B. (2010). *Google I/O 2010 - A JIT Compiler for Android's Dalvik VM*. [online] YouTube. Available at: https://www.youtube.com/watch?v=Ls0tM-c4Vfo [Accessed 1 May 2015].

2.3 Android Runtime

Android version 4.4 provided users with the option to change their runtime from the Dalvik Virtual Machine to an experimental version of Android Runtime. The later version, Android 5.0, set Android Runtime as the default runtime, removing the Dalvik Virtual Machine. The new runtime works to interpret and compile the same Dalvik executable files that the Dalvik Virtual Machine was able to handle. This continuity is important because it allows any application, in theory, to run on the new runtime without the developer having to make any changes to it. The large difference between the two runtimes was the change from the Just-In-Time compiler to the Ahead-of-Time compiler. The change causes the bytecode to only be compiled when the application is first installed, instead of it occurring every time it was opened. After installation and compilation, the native code that has been produced is then saved to the device to be executed when the application is opened[21]. This brings about the advantage of applications opening faster; however, due to the fact they are being stored in native code, they take up more storage space.

With the use of Android Runtime, .odex files are no longer used and instead they have been replaced by ELF executable files. Due to this change the kernel is now able to support kernel same-page merging[22]. Kernel same-page merging allows processes which have identical memory pages (data stored in RAM) to share them, avoiding duplicates and decreasing the memory used. This does however increase the processor use so the increase in performance may not be as apparent.[23]

[21] Frumusanu, A. (2014). *A Closer Look at Android Runtime (ART) in Android L*. [online] Anandtech.com. Available at: http://anandtech.com/show/8231/a-closer-look-at-Android-runtime-art-in-Android-l/ [Accessed 1 May 2015].

[22] Ibid

[23] Rouse, M. (2014). *What is KSM (kernel samepage merging)? - Definition from WhatIs.com*. [online] SearchServerVirtualization. Available at: http://searchservervirtualization.techtarget.com/definition/KSM-kernel-samepage-merging [Accessed 3 Sep. 2015].

3. Comparison

Making the comparison requires two aspects to be studied; the first is the theoretical comparison of the differences between the two runtimes. This allows us to see in theory how one runtime should improve upon the other. Then the practical comparison can explain whether the system experiences these theoretical improvements.

3.1 Theoretical Comparison

One of the ways that Android Runtime improves upon the Dalvik Virtual Machine is the advancement in garbage collection. Garbage collection is a process which aims to keep RAM free so that the device can have many applications running well together. The process works to avoid memory leaks (a process filling up RAM unnecessarily[24]) by finding all live objects, useful pieces of data, in a running application and marking the position in memory of all other objects as empty, subsequently removing all unreachable but not deleted objects. This allows developers to not focus on object allocation and removal because the garbage collector can, in theory, remove all these inaccessible objects. The problem arises due to the method of distinguishing between the live and dead objects. All objects have garbage collection roots which involve local variables, active Java threads, static variables and JNI references and if these roots are accessible by the code, then all other objects stemmed from those roots, must be accessible. To find the dead objects, the virtual machine uses a mark-and-sweep algorithm that runs through the program to find the garbage collection roots[25]. The speed of garbage collection relies on the number of live objects there are, if the application has many live objects then the application could pause for a few seconds. This interruption occurs because, to find all live garbage collection roots and ensure integrity, the application has to be paused for the duration of the garbage collection. As this can be seconds long there is a big impact on the user's

[24] Rouse, M. (2005). *What is memory leak? - Definition from WhatIs.com*. [online] SearchWindowsServer. Available at: http://searchwindowsserver.techtarget.com/definition/memory-leak [Accessed 6 Sep. 2015].
[25] Dynatrace.com, (n.d.). *How Garbage Collection Works - Java Enterprise Performance Book | Dynatrace*. [online] Available at: http://www.dynatrace.com/en/Javabook/how-garbage-collection-works.html [Accessed 1 May 2015].

experience because loading times would increase and many frames would be dropped leading to stuttering[26].

The first method Android Runtime uses to improve the garbage collection is the use of only one garbage collection pause instead of two[27]. When running Dalvik, if an application requires allocation of memory that heap (the amount of memory that an application is allocated by the Java virtual machine[28]) cannot accommodate, the garbage collector would be enabled with two pauses[29]. The garbage collector in Android Runtime, however, only requires one pause, which reduces the amount of time the application has to be halted for, therefore decreasing the amount of dropped frames and increasing the performance of the application.

The second method is the use of parallel processing whilst the garbage collection is being completed[30]. This method allows for the use of multiple central processing unit cores to execute the mark-and-sweep algorithm for multiple threads at once, this reduces the time that the application has to be paused for, which increases the performance of the applications[31].

Android Runtime includes the new Ahead-of-Time compiler replacing the older Just-in-Time compiler. This replacement causes all compilation to native code to be carried out at the time of installation. This native code is then saved and used from then onwards. This has a few advantages and disadvantages associated with it. Due to the native code being saved onto the device, the application doesn't need to waste time compiling the code every time it is opened, thus improving the start-up

[26] Dynatrace.com, (n.d.). *The Impact of Garbage Collection on application performance - Java Enterprise Performance Book | Dynatrace*. [online] Available at: http://www.dynatrace.com/en/Javabook/impact-of-garbage-collection-on-performance.html [Accessed 1 May 2015].

[27] Source.Android.com, (n.d.). *ART and Dalvik | Android Developers*. [online] Available at: https://source.Android.com/devices/tech/dalvik/ [Accessed 1 May 2015].

[28] Azulsystems.com, (n.d.). *How Java Garbage Collection Impacts Maximum Heap Size*. [online] Available at: http://www.azulsystems.com/technology/Java-heap-size [Accessed 17 Sep. 2015].

[29] Frumusanu, A. (2014). *A Closer Look at Android Runtime (ART) in Android L*. [online] Anandtech.com. Available at: http://anandtech.com/show/8231/a-closer-look-at-Android-runtime-art-in-Android-l/2 [Accessed 3 May 2015].

[30] Dynatrace.com, (n.d.). *The Impact of Garbage Collection on application performance - Java Enterprise Performance Book | Dynatrace*. [online] Available at: http://www.dynatrace.com/en/Javabook/impact-of-garbage-collection-on-performance.html [Accessed 1 May 2015].

[31] Dynatrace.com, (n.d.). *Reducing Garbage Collection Pause time - Java Enterprise Performance Book | Dynatrace*. [online] Available at: http://www.dynatrace.com/en/javabook/reduce-garbage-collection-pause-time.html [Accessed 1 May 2015].

performance. Another benefit that we can see from the compilation happening once, is the reduction in the usage of the computer processing unit. When the application is compiled it requires power from the battery to run the central processing unit, if this only happens once then less power is needed so the battery is drained less. The longer a device's battery lasts, the more usage will be achieved by the user, thus increasing their productivity and performance. However, a disadvantage is the increase in storage size of every application, which is one of the reasons the process has not been adopted before now. The native code of an application is much larger in terms of storage size than the bytecode which was previously saved to the device[32]. The reason for this is because each bytecode represents multiple instructions, thus decreasing the repetition that is found in the native code. The developers aim was that this increase should not be more than 10-20% of the entire application size, due to only a portion of that data being executable code[33]. This provides every application with a larger storage requirement, in turn limiting the number of applications and other data the user could save to their device. This is only possible due to the increased storage capacity many devices have now, phones are now often released with 32GB or more of storage[34] whereas the first android phone (HTC Dream) had only 256MB of data[35]. Another small disadvantage can be seen due to the application being compiled when it is installed, this therefore increases the time it takes to install it. However, as this only happens once in the background, it shouldn't affect the user's performance as much[36]. This increased installation time does, however, come with an added benefit proving how

[32] Dynatrace.com, (n.d.). *How Garbage Collection Works - Java Enterprise Performance Book | Dynatrace*. [online] Available at: http://www.dynatrace.com/en/Javabook/how-garbage-collection-works.html [Accessed 1 May 2015].
[33] Toombs, C. (2013). *Meet ART, Part 1: The New Super-Fast Android Runtime Google Has Been Working On In Secret For Over 2 Years Debuts In KitKat*. [online] Android Police. Available at: http://www.Androidpolice.com/2013/11/06/meet-art-part-1-the-new-super-fast-Android-runtime-Google-has-been-working-on-in-secret-for-over-2-years-debuts-in-kitkat/ [Accessed 3 May 2015].
[34] Samsung Electronics America, (2015). *Get the Latest on the New Samsung Galaxy S 6 &amp; 6 Edge | Samsung*. [online] Available at: http://www.samsung.com/us/explore/galaxy-s-6-features-and-specs/ [Accessed 16 Sep. 2015].
[35] Gsmarena.com, (n.d.). *HTC Dream - Full phone specifications*. [online] Available at: http://www.gsmarena.com/htc_dream-2665.php [Accessed 16 Sep. 2015].
[36] Dynatrace.com, (n.d.). *Reducing Garbage Collection Pause time - Java Enterprise Performance Book | Dynatrace*. [online] Available at: http://www.dynatrace.com/en/javabook/reduce-garbage-collection-pause-time.html [Accessed 1 May 2015].

Ahead-of-Time compilation can improve upon Just-in-Time compilation. The compilation occurring only once, and at installation, allows for a better optimised compilation of the entire application to be completed, creating better performing native code.

Kernel same-page merging, as previously discussed, allows applications to share memory pages when different processes have identical pages. This, in theory, can allow for a great reduction in memory use, but can increase the use of the central processing unit. As performance takes into account both memory management and battery life, the increase in performance that should occur may be minimal.

3.2 Practical Comparison

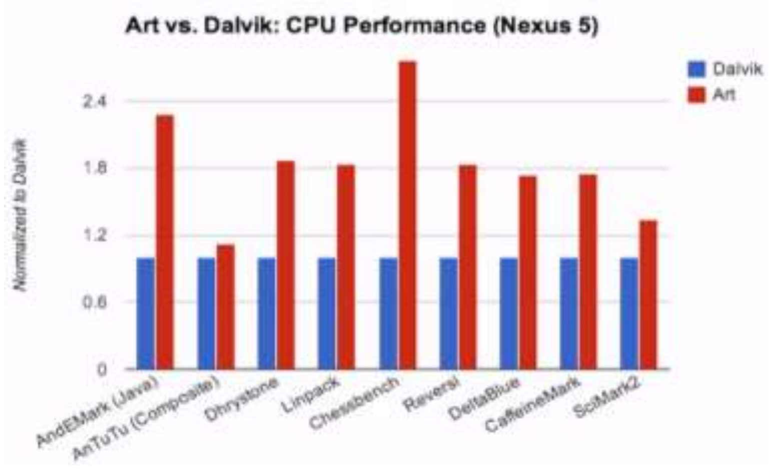
The first test to be completed is how much storage space an application takes up, first when using Dalvik as the runtime and then secondly when the runtime is Android Runtime. To make this test a range of applications will be installed and the runtime will be changed noting the storage space used by each application. It is expected that there will be an increase in storage size for all applications due to the code being compiled to native code when the app is installed and the native code taking up more storage space than the previous bytecode. This increase in storage space is expected due to the larger footprint native code has over the previously stored bytecode. Figure 3.2.1 presents the results of the test.

Figure 3.2.1 – A table showing the storage size of applications when using the Dalvik Virtual Machine and Android Runtime. All applications had never been run when the measurements were taken.

Name of application	Dalvik Virtual Machine	Android Runtime	Percentage Increase
Dictionary	15.71 MB	28.41 MB	81%
Score Center	3.57 MB	6.67 MB	87%
Temple Run	41.21 MB	43.05 MB	4%
Gmail	20.17 MB	31.48 MB	56%
Hangouts	42.06 MB	61.43 MB	46%
Twitter	30.21 MB	44.27 MB	47%
Colour Switch	30.36 MB	43.12 MB	42%

It can be seen that there is a large increase in storage size between the two runtimes. The largest increase in application size occurred for *Score Center* which increased by 87%. The smallest increase was 4% that occurred to *Temple Run*. The results show that there is a large variation in how much different applications were affected by the change. One reason for this, highlighted before, was that only a section of the stored data is the executable code that can be compiled to native code, therefore different applications will experience a different increase. Changing runtime should result in less stuttering of applications and shorter loading times due to the new method of garbage collection. The shorter the pause, the better the user experience and subsequently the performance because less frames are dropped. An experiment was carried out using the FIFA application and measurements were made on the length of time the application was paused for garbage collection. When running the Dalvik runtime the garbage collector ran 9 times causing the application to pause for a total of 603ms with 214 dropped frames. The same system running Android Runtime called the garbage collector 4 times in the foreground and twice in the background. The length of time the application was paused was reduced massively to 12.364ms with only 63 frames dropped in total[37]. This results in a better user experience and performance as the application would pause less, drop less frames and run processes quicker.

Figure 3.2.2[38] – A bar chart displaying a comparison between the results of different benchmarks when running Android Runtime and the Dalvik Virtual Machine.



[37] Frumusanu, A. (2014). *A Closer Look at Android Runtime (ART) in Android L*. [online] Anandtech.com. Available at: http://anandtech.com/show/8231/a-closer-look-at-Android-runtime-art-in-Android-l/2 [Accessed 4 May 2015].

If figure 3.2.2[38] is studied, the performance improvement of Android Runtime can be seen through the use of multiple benchmarks. These benchmarks were carried out by Google to prove the increase in performance they claimed would occur to be present. All benchmarks show that there has been a great improvement from changing the runtime from the Dalvik Virtual Machine to Android Runtime. The smallest increase in performance was experienced by the AnTuTu benchmark, however this is explained by the nature of the benchmark being a composite benchmark. This means it measures many different factors including .dex code and native code; subsequently only an increase in performance should be expected on the .dex code sections[39]. The largest increase in the central processing unit performance can be seen using the Chessbench score, which is more than double the Dalvik score. This is impressive as it is the most accurate measure due to it being the most similar to how a developer writes applications for the current system.

The start-up time of applications is another key improvement when using Android Runtime over the Dalvik Virtual Machine. This is, however, something that is very hard to accurately measure due to it varying from other external variables such as the currently running processes. To test the start-up times for different applications a variety of them were opened using both runtimes and the time taken was recorded.

Figure 3.2.3 – A table showing the start-up time of different applications when using both runtimes

Name of Application	Dalvik Virtual Machine (Seconds)	Android Runtime (Seconds)
Dictionary	1.7	1.6
Score Center	0.9	0.9
Temple Run	3.8	3.5
Gmail	2.2	2.2
Hangouts	1.1	1.0
Twitter	2.5	1.9
Colour Switch	3.2	3.2

[38] Ghuloum, A., Carlstrom, B. and Rogers, I. (2014). *Google I/O 2014 - The ART Runtime*. [online] YouTube. Available at: https://www.youtube.com/watch?v=EBlTzQsUoOw?t=13m27s [Accessed 6 May 2015].
[39] Antutu.com, (2016). *Ranking - AnTuTu Benchmark -- Know Your Android Better*. [online] Available at: http://www.antutu.com/en/Ranking.shtml [Accessed 3 Oct. 2015].

The results show that most applications experienced a decrease in the amount of time it took for them to load when using Android Runtime. The applications that didn't experience a decrease maintained their opening time showing that Android Runtime didn't increase the start-up time for any of the applications tested. The application which experienced the largest decrease in opening time was Temple Run with a decrease of 0.3 seconds. This is surprising due the storage size of this application, found at the beginning of the comparison, having the smallest percentage increase, implying that the majority of the application was not .dex code; which would have been affected by the change in runtime. Therefore, this is promising as other applications should experience a similar or greater opening speed increase. This decrease is promising as it would speed up the use of the phone thus directly improving the performance for the user.

The applications are compiled to native code at the point of installation when using the Ahead-of-Time compiler found in Android Runtime. This should mean that the time taken for installation by applications should increase when using the new runtime. To test this the application installation time was measured for a variety of applications when using both runtimes and the results can be seen below.

Figure 3.2.4 – A table showing the time taken by different applications to install on both runtimes

Name of Application	Dalvik Virtual Machine (Seconds)	Android Runtime (Seconds)
Colour Switch	17	57
Dictionary	15	42
Twitter	19	37
Score Center	6	12
Temple Run	16	27

It can be seen that all applications experienced a large increase in the time it took for them to install when using Android Runtime. The largest increase was experienced by Colour switch with an increase of 40 seconds which made it 3.35 times its original length. This shows that the increase in installation is a large issue however its overall effect on the user is debatable as the installation only occurs once.

4. Conclusion

This essay aimed to examine the extent to which Android Runtime improved upon the Dalvik virtual machine in terms of performance. For the purpose of the essay, performance was measured in terms of speed, memory use, storage size and battery life.

The introduction of Android Runtime has affected the speed that the system runs in many ways; for example, due to the reduced processing required as the application starts. The application is compiled to native code at installation and therefore the application can open quickly without it having to be compiled. This does however increase the time it takes to install an application and this could deter consumers from downloading as many applications. The increase in speed was present when measuring the opening times of applications on both runtimes; however, the increase in time taken when applications were installed was also evident and great. Garbage collection improvements affect both speed and memory use as the new garbage collection process aims to free memory at a greater speed using parallel processing and only one pause. This was seen to be effective in practical testing which means that less frames should be dropped improving the user experience. Android Runtime makes a big impact on the storage space of applications due to the Ahead-of-Time compiler. All tested applications were affected with an increase in storage size but the magnitude of the problem ranged greatly. This negatively affects the user as they will be able to install less applications and store less music, photos and other files. Finally, the battery is drained every time the processor is enabled and therefore using the processor less by compiling the application only once at installation should have a positive effect on the battery life of a device. This increase is, however, very small and is possibly offset by the increased processing of kernel same-page merging.

Android Runtime can therefore be seen to have many benefits in terms of speed and memory use, however it does lead to problems with increased storage usage. Due to advancements in storage technology for devices, this issue isn’t as important as it used to be and therefore the main impact this change will have on the user is in relation to speed. A faster device allows the user to get more done in a shorter amount of time, which increases productivity and user satisfaction. This information leads to the conclusion that users of Android 4.4 should choose to use Android Runtime as the default runtime and, if it is available to them, they should update the phones to Android 5.0. This will ultimately increase the performance of their device and when referring back to users of devices like the Keepod, where resources are limited, this could greatly increase what the users are capable of achieving.

5. Bibliography

1. Amadeo, R. (2014). *The history of Android*. [online] Ars Technica. Available at: http://arstechnica.com/gadgets/2014/06/building-Android-a-40000-word-history-of-Googles-mobile-os/ [Accessed 13 Apr. 2015].
2. Antutu.com, (2016). *Ranking - AnTuTu Benchmark -- Know Your Android Better*. [online] Available at: http://www.antutu.com/en/Ranking.shtml [Accessed 3 Oct. 2015].
3. Azulsystems.com, (n.d.). *How Java Garbage Collection Impacts Maximum Heap Size*. [online] Available at: http://www.azulsystems.com/technology/Java-heap-size [Accessed 17 Sep. 2015].
4. Bornstein, D. (2010). *Google I/O 2008 - Dalvik Virtual Machine Internals*. [online] YouTube. Available at: https://www.youtube.com/watch?v=ptjedOZEXPM [Accessed 30 Apr. 2015].
5. Caprio, G. (2006). *Virtual Machines: Virtualization vs. Emulation*. [online] Griffincaprio.com. Available at: http://www.griffincaprio.com/blog/2006/08/virtual-machines-virtualization-vs-emulation.html [Accessed 17 Sep. 2015].
6. Centos.org, (n.d.). *5.1. What is Swap Space?*. [online] Available at: https://www.centos.org/docs/5/html/5.2/Deployment_Guide/s1-swap-what-is.html [Accessed 17 Sep. 2015].
7. Cheng, B. and Buzbee, B. (2010). *Google I/O 2010 - A JIT Compiler for Android's Dalvik VM*. [online] YouTube. Available at: https://www.youtube.com/watch?v=Ls0tM-c4Vfo [Accessed 1 May 2015].
8. Developer.Android.com, (2015). *Dashboards | Android Developers*. [online] Available at: https://developer.Android.com/about/dashboards/index.html [Accessed 08 May 2015].
9. Differencebetween.info, (n.d.). *Difference between Virtual Memory and Swap Memory*. [online] Available at: http://www.differencebetween.info/difference-between-virtual-memory-and-swap-memory [Accessed 8 May 2015].
10. Dynatrace.com, (n.d.). *How Garbage Collection Works - Java Enterprise Performance Book | Dynatrace*. [online] Available at: http://www.dynatrace.com/en/Javabook/how-garbage-collection-works.html [Accessed 1 May 2015].
11. Dynatrace.com, (n.d.). *Reducing Garbage Collection Pause time - Java Enterprise Performance Book | Dynatrace*. [online] Available at: http://www.dynatrace.com/en/javabook/reduce-garbage-collection-pause-time.html [Accessed 1 May 2015].
12. Dynatrace.com, (n.d.). *The Impact of Garbage Collection on application performance - Java Enterprise Performance Book | Dynatrace*. [online] Available at: http://www.dynatrace.com/en/Javabook/impact-of-garbage-collection-on-performance.html [Accessed 1 May 2015].
13. Epec Engineered Technologies, (2014). *New Battery Technology Developments*. [online] Slideshare.net. Available at: http://www.slideshare.net/epectec/new-battery-trends [Accessed 17 Sep. 2015].
14. forensic blog, (2012). *Comparison of Dalvik and Java Bytecode*. [online] Available at: http://forensics.spreitzenbarth.de/2012/08/27/comparison-of-dalvik-and-Java-bytecode/ [Accessed 1 May 2015].
15. Frumusanu, A. (2014). *A Closer Look at Android Runtime (ART) in Android L*. [online] Anandtech.com. Available at: http://anandtech.com/show/8231/a-closer-look-at-Android-runtime-art-in-Android-l/ [Accessed 1 May 2015].
16. Frumusanu, A. (2014). *A Closer Look at Android Runtime (ART) in Android L*. [online] Anandtech.com. Available at: http://anandtech.com/show/8231/a-closer-look-at-Android-runtime-art-in-Android-l/2 [Accessed 3 May 2015].
17. Frumusanu, A. (2014). *A Closer Look at Android Runtime (ART) in Android L*. [online] Anandtech.com. Available at: http://anandtech.com/show/8231/a-closer-look-at-Android-runtime-art-in-Android-l/2 [Accessed 4 May 2015].
18. Ghuloum, A., Carlstrom, B. and Rogers, I. (2014). *Google I/O 2014 - The ART Runtime*. [online] YouTube. Available at: https://www.youtube.com/watch?v=EBlTzQsUoOw?t=13m27s [Accessed 6 May 2015].
19. Gsmarena.com, (n.d.). *HTC Dream - Full phone specifications*. [online] Available at: http://www.gsmarena.com/htc_dream-2665.php [Accessed 16 Sep. 2015].
20. Hornyak, T. (2014). *Android grabs record 85 percent smartphone share*. [online] PCWorld. Available at: http://www.pcworld.com/article/2460020/Android-grabs-record-85-percent-smartphone-share.html [Accessed 13 Apr. 2015].

21. Mooreslaw.org, (n.d.). *Moore's Law*. [online] Available at: http://www.mooreslaw.org/ [Accessed 30 Apr. 2015].
22. Reilly, D. (2006). *Inside Java : The Java Virtual Machine*. [online] Javacoffeebreak.com. Available at: http://www.Javacoffeebreak.com/articles/inside_Java/insideJava-jan99.html [Accessed 17 Sep. 2015].
23. Rouse, M. (2005). *RAM (random access memory)*. [online] SearchStorage. Available at: http://searchstorage.techtarget.com/definition/RAM-random-access-memory [Accessed 17 Sep. 2015].
24. Rouse, M. (2005). *What is memory leak? - Definition from WhatIs.com*. [online] SearchWindowsServer. Available at: http://searchwindowsserver.techtarget.com/definition/memory-leak [Accessed 6 Sep. 2015].
25. Rouse, M. (2006). *What is Java virtual machine (JVM)? - Definition from WhatIs.com*. [online] SearchSOA. Available at: http://searchsoa.techtarget.com/definition/Java-virtual-machine [Accessed 30 Apr. 2015].
26. Rouse, M. (2014). *What is KSM (kernel samepage merging)? - Definition from WhatIs.com*. [online] SearchServerVirtualization. Available at: http://searchservervirtualization.techtarget.com/definition/KSM-kernel-samepage-merging [Accessed 3 Sep. 2015].
27. Samsung Electronics America, (2015). *Get the Latest on the New Samsung Galaxy S 6 &amp; 6 Edge | Samsung*. [online] Available at: http://www.samsung.com/us/explore/galaxy-s-6-features-and-specs/ [Accessed 16 Sep. 2015].
28. Simmons, D. (2014). *Keepod: Can a $7 stick provide billions computer access? - BBC News*. [online] BBC News. Available at: http://www.bbc.co.uk/news/technology-27346567 [Accessed 4 May 2015].
29. Smith, J. and Nair, R. (2005). The architecture of virtual machines. *Computer*, [online] 38(5), pp.32-38. Available at: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1430629 [Accessed 17 Sep. 2015].
30. Source.Android.com, (n.d.). *ART and Dalvik | Android Developers*. [online] Available at: https://source.Android.com/devices/tech/dalvik/ [Accessed 1 May 2015].
31. Toombs, C. (2013). *Meet ART, Part 1: The New Super-Fast Android Runtime Google Has Been Working On In Secret For Over 2 Years Debuts In KitKat*. [online] Android Police. Available at: http://www.Androidpolice.com/2013/11/06/meet-art-part-1-the-new-super-fast-Android-runtime-Google-has-been-working-on-in-secret-for-over-2-years-debuts-in-kitkat/ [Accessed 3 May 2015].
32. Venners, B. (1996). *Bytecode basics*. [online] JavaWorld. Available at: http://www.Javaworld.com/article/2077233/core-Java/bytecode-basics.html [Accessed 17 Sep. 2015].