IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 2, February 2017

Android OS for Embedded Real-Time Systems

Smita Sontakke[1], Jagruti Thombare[2], Pournima Lad[3]

Student, Computer Engineering, Bharati Vidyapeeth Institute of Technology, Navi Mumbai, India[1, 2]

Professor, Computer Engineering, Bharati Vidyapeeth Institute of Technology, Navi Mumbai, India[3]

Abstract: Since Android is recently publicly introduced, Android has taken the interest from companies and the common Audience. This software platform has been constantly enhanced either in term of features or supported hardware; these are extended to new types of devices different from the originally contracted mobile ones. However, there is a feature that has not been researched yet - its real-time capabilities. This paper proposes to remove this gap and provide a basis for discussion on the appropriateness of Android in order to be used in Open Real-Time environments. By analysing the software platform, with the main goal of the virtual machine and its underlying operating system environments, we are able to point out its current restrictions. And using this, we are able to provide a hint on different angel of directions in order to make Android suitable for the environments. It is our position that Android may contribute a suitable architecture for real-time embedded systems, but the real-time community should address its restrictions in a joint effort at all of the platform layers.

Keywords: Android, Open Real-Time Systems, Embedded Systems, Suitability.

I. INTRODUCTION

Android was made available for the public in the year 2008. Being considered a new technology, due to the fact that it is still being essentially improved and upgraded in terms of features or firmware, Android is obtaining strength both in the mobile industry and in other industries with different hardware architectures. The growing interest from the industry comes from two core aspects: its open-source nature and its architectural model. Being an open-source project, allows Android to be fully imagined and understood, which empowers feature comprehension, bug fixing, further progress regarding new functionalities. Linux kernel-based architecture also includes the use of Linux to the mobile industry, granting access to take advantage of the knowledge and is featured by Linux. Both of these aspects make Android an achieving target to be used in other type of environments. Another aspect that is necessary to consider when using Android is, its own Virtual Machine environment. In the use of Virtual Machine environment its feature is based on java that orders Android applications with both its advantages. Nevertheless, there are features which have not been tested yet, as for instance the correctness of the platform to be used in Open Real-Time environments. Taking into concern, works developed before such as [1] [2] regarding the Linux kernel or VM environment, there are the possibilities of proposing temporal guarantees associated with Quality of Service guarantees in each of the preceding layers, in a way that integration may be achieved, fulfilling the physical constraints established by the applications.

This integration may be helpful for multimedia applications requiring specific machine resources that need to be guaranteed in an advanced and timely manner. Therefore, taking advantage of the real-time efficiency and resource that are optimized and are applied by the platform.

II. HISTORY

The versions 2.6.23, the standard Linux kernel handles the Fair Scheduler, which gives fairness in the way that CPU time is given to tasks. These balancing guarantees that all the tasks will have the same CPU share and each time that is unfairness is verified.
Android uses its own VM named “Dalvik”, which was specially developed for hands free mobile devices and contains memory consumption, battery power saving and low frequency of all CPU. It is based on the Linux kernel for the core operating system such as memory management and scheduling and hence, gives the disadvantage of not taking any guarantees into deliberation. The thesis in this paper is a chunk of the Cooperative Embedded Systems project [3], which aspires at the implementation of a QoSaware framework; determine in [4], to be used in an open and dynamic cooperative environment. Due to the environments nature, the framework should support resources before and guarantee that the real-time execution restrain imposed by the applications are satisfied.

In the outlook of the project, there was the need of appraising Android as one of the target solutions to be used for the framework’s implementation. As a result of this assessment, this paper addresses the probable of Android and the implementation advice that can be accepted in order to make it available in Open Real-Time environments. However, our focus is on to soften the real-time applications and accordingly, hard-real time applications were not examined in our evaluation.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 2, February 2017

III. ANDROID'S ARCHITECTURE

Android is open-source software architecture conducted by the Open Handset Alliance [5], a group of 71 technology and mobile companies whose motto is to provide a mobile software platform. The Android platform comprises of an operating system, middleware and applications. As for the features, Android consolidates the features found in any mobile device platform, such as application framework reusing, integrated browser, optimised graphics, media support, network technologies, etc. The Android architecture, illustrated in Figure 1, is made by five layers: Applications, Application Framework, Libraries, Android Runtime and finally the Linux kernel. The uppermost layer, the Applications layer, provides the core set of applications that are frequently offered out of the box with any mobile device. The Application Framework layer gives the framework Application Programming Interfaces used by the applications working on the uppermost layer.

Apart from the Application Programming Interfaces, there is a set of services that allows the access to Android's main aspects such as graphical components, event managers and activity managers. Below the Application Framework layer, there is another layer having two important parts: Libraries and the Android Runtime. The libraries provide core features to the applications. Among all the libraries given, the important library are lib c, the standard C system library accommodate for embedded Linux-based devices; the Media Libraries, which support recording of various audio and playback formats; a lightweight relational database engine, Graphics Engines and 3D libraries Regarding the Android Runtime, Dalvik [6] was designed from blemish and it is specifically aimed for memory related and CPU related devices. It runs Java applications and dislikes the standard Java Virtual Machines, which are stack-based; Dalvik is a multiple register-based machine. The version 2.6 of Linux kernel is the endmost layer and is a hardware abstraction layer that allows the intercommunication of the upper layers with the hardware layers via device drivers.

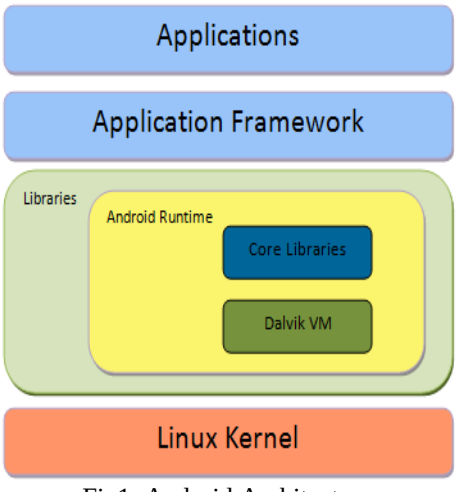

Fig1. Android Architecture

IV. SUITABILITY OF ANDROID FOR OPEN REAL - TIME SYSTEMS

This part discusses the appropriateness of Android for open embedded real-time systems; inquire its architecture internals and mark out its current limitations. Android was appraised considering the following topics: its VM environment, the underlying Linux kernel, and its resource management comp entice. Dalvik VM is able of running multiple independent processes.

Therefore, each Android application is graphed to a Linux process and able to use an inter-process communication mechanism, based on Open-Binder [7], to communicate with other processes in the system. The capability of separating each process is given by Android's architectural model. During boot time, there is a process culpable for starting up the Android's runtime, which implies the start-up of the VM itself. Built in to this step, there is a VM process, the Zygote, culpable for the pre-initialisation and pre-loading of the Android's classes that will be the applications. Later, the Zygote opens a socket that takes commands from the application framework whenever a new Android application has begun. This will affect the Zygote to be angled and create a child process which will then be the target application.

This model is presented in Figure 2. The approach is favourable for the system as, and it is possible to save RAM to speed up each application start up process. Android applications provide the synchronisation mechanisms known to the Java community. The VM controls all the threads an internal structure where all the created threads are graphed. The Graphical Collector will only run when all the threads attributing to a single process are appended, in order to avoid in consistent states. The GCs have the hard task of handling dynamic memory management, as they are caught hold for deal locating the memory allocated by objects that are no longer needed by the applications.

Relating to Android's garbage collection process, as the processes run individually from other processes and each process has its own heap and a shared heap – the Zygote's heap- Android runs independent illustrations of GCs with a motive to collect memory that is not going to be used anymore. Thus, each process heap results into garbage that is collected independently, through the usage of parallel mark bits that specify which objects

This process is particularly useful in Android due to the Zygote's shared heap, which in this case is kept unattained by the GC to put memory to a better use.

At present, the Linux kernel suggests two classes of real-time scheduling, as a part of the agreement with the POSIX standard [8], SCHED RR and SCHED FIFO. SCHED FIFO used for a first-in, first-out scheduling and SCHED RR for a round robin scheduling. These scheduling methods have a huge effect on system's performance if irrelevant programming applies. However, many of the tasks are scheduled with SCHED OTHER class that is a non real-time scheduling method. The task scheduling plays one of the most important roles regarding

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 2, February 2017

the real-time features granted by a particular system. Two scheduling real-time classes are currently limited to Linux real-Time system; they are based on priority scheduling. More important aspect to be studied in the assessment is that most of the tasks are scheduled by CFS. Even though CFS tries to optimise the time, a task is waiting for CPU time. At the kernel level, with the exception of the CPU and memory, all the remaining system's hardware is accessed via device drivers, in order to perform its operations and control the resources' status.

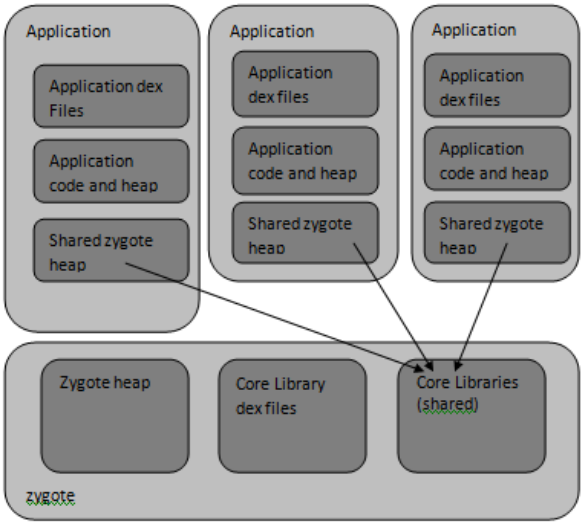

Fig2.Zygote Heap

V. POSSIBLE DIRECTIONS

This part explains the four possible ways to consolidate the required real-time behaviour into the Android architecture. The first approach includes the restoration of the Linux operating system by one that gives real-time features and, at the parallel time, it estimates the enclosure of the real-time Virtual Machine. The second approach has the Android standard architecture by proposing the extension of Dalvik among other things that change over the standard operating system through the real-time Linux-based operating system. The third approach regains the Linux operating system for a Linux real-time version plus real-time applications use the kernel directly. At last, the fourth approach requires the expansion of a real-time hypervisor. Regarding the first approach, interpreted in Figure 3, this approach changes the standard Linux kernel with a real-time operating system. This adjustment recommended predictability and determinism in the Android architecture. As a result, it is probable to suggest new dynamic real-time scheduling policies over the usage of scheduling classes; predict priority inversion and to contain superior resource management methods.

The second change, within the first approach, is the extension of the real-time Java Virtual Machine. This change is studied profitable as, it is apparently to contain bounded memory management; real-time scheduling within the VM, rely on the selected explanation; inferior synchronisation mechanisms and lastly end preference inversion. This improvement is considered the most authoritative in gaining the required behaviour at the VM level.

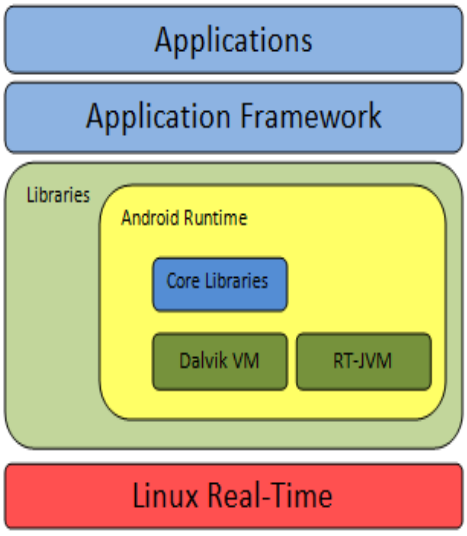

Fig3.Android full Real-Time

The more benefit given from this approach is that it is not important to maintain the release cycles of Android, even if some integration problem may occur between the Virtual Machine and the kernel. The brunt of introducing a new VM in the system is similar to the fact that all the Android must be implemented as well as dex in the interpreter. Apart from this, other challenges may be such as the integration between both VMs.

The second recommended approach, given in Figure 4, also presents changes in the architecture both in the operating system and virtual machine environments. As for the operating system layer, the advantages and disadvantages shown in the first approach are compared equal, as the principle behind it is the equal.

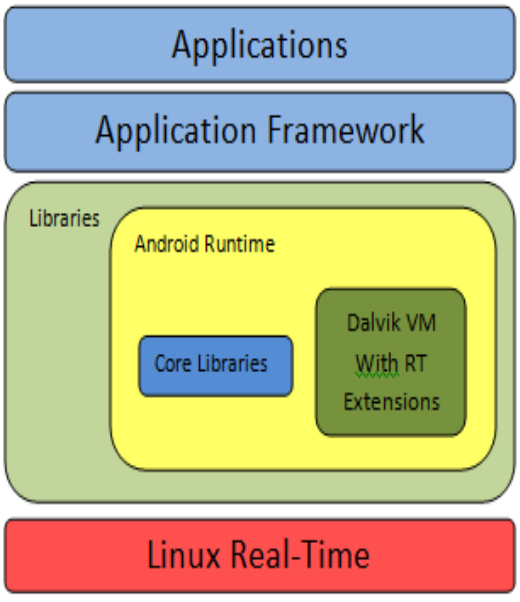

Fig4.Android Extended

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 2, February 2017

The third recommended approach, represents in Figure 6, it is also based in Linux real-time. This approach takes benefit of the native environment, where it is probable to open the real-time applications directly over the operating system. This can be beneficial for applications that do not required the VM environment, which means that a minimal effort will be required for integration, while having the same expected behaviour. On the other hand, applications that required a VM environment will not help from the real-time services of the hidden operating system.

VI. CONCLUSION

At first glance, Android looks like a potential target for real-time environments and, there are various industry targets that would help from an architecture with such facility. Taking this into consideration, this paper introduced the assessment of the Android platform to be used as a real-time system. By focusing on the importance parts of the system it was probable to uncover the limitations and then, to current four probable directions that may be followed to add real-time behaviour of the system.

Android was constructed to deliver the mobile industry purposes and that fact has an impact on the way that the architecture might be used. However, with some effort, as proven by the presented approaches, it is possible to have the desired Real-Time behaviour on any Android device.

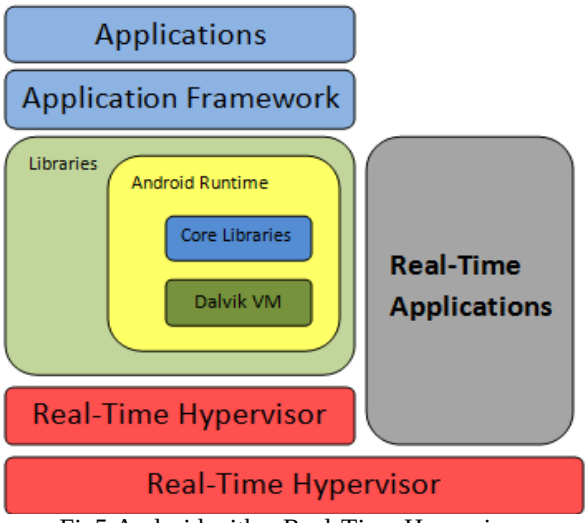

Fig5.Android with a Real-Time Hypervisor

Finally, the fourth approach depicted in Figure 5, employs a real-time hypervisor that is capable of running Android as a guest operating system in one of the partitions and real-time applications in another partition, in a parallel manner. This approach is similar to the approach taken by the majority of the current real-time Linux solutions, such as RTLinux [9] or RTAI [10]. These systems are able to run real-time applications in parallel to the Linux kernel, where the real-time tasks have higher priority than the Linux kernel tasks, which means that hard real-time can be used.

ACKNOWLEDGEMENT

Thanks to our guide, and our college management for providing the resources and helping us in all the possible ways. We also thank readers of this journal for showing interest in this topic and contributing towards the enhancement of this topic as well.

REFERENCES

[1] RTMACH, "Linux/rk," Mar. 2010. [Online]. Available: http://www.cs.cmu.edu/~rajkumar/linux-rk.html
[2] Corsaro, "jrate home page," Mar. 2010. [Online]. Available: http://jrate.sourceforge.net/
[3] CooperatES, "Home page," Jan. 2010. [Online]. Available: http://www.cister.isep.ipp.pt/projects/cooperates/
[4] L. Nogueira and L. M. Pinho, "Time-bounded distributed qosaware service configuration in heterogeneous cooperative environments," Journal of Parallel and Distributed Computing, vol. 69, no. 6, pp. 491–507, June 2009. 69
[5] O. H. Alliance, "Home page," Jun. 2010. [Online]. Available: http://www.openhandsetalliance.com/
[6] D. Bornstein, "Dalvik vm internals," Mar. 2010. [Online]. Availablehttp://sites.google.com/site/io/dalvik-vm-internals
[7] P. Inc., "Openbinder 1.0," Mar. 2010. [Online]. Available: http://www.angryredplanet.com/~hackbod/openbinder/
[8] IEEE, "Ieee standard 1003.1," Mar. 2010. [Online]. Available: http://www.opengroup.org/onlinepubs/009695399/
[9] W. R. Systems, "Real-time Linux," Jun. 2010. [Online]. Available: http://www.rtlinuxfree.com/
[10] P. d. M. Dipartimento di Ingegneria Aerospaziale, "Real-time application interface for Linux," Jun. 2010. [Online]. Available: https://www.rtai.org/