ÁREA DE INFORMÁTICA

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - ADS

ACÁCIO ALKMIN DE ALMEIDA CASSIANO

GUILHERME OLIVEIRA GUERRA

ANÁLISE DO DESEMPENHO DAS TÉCNICAS DE PROGRAMAÇÃO RECOMENDADAS PARA O ANDROID NAS MÁQUINAS VIRTUAIS ART E DALVIK

TRABALHO DE CONCLUSÃO DE CURSO - TCC

CARAGUATATUBA

2015

G934a Guerra, Guilherme Oliveira

Análise do desempenho das técnicas de programação recomendadas para o Android nas máquinas virtuais ART e Dalvik / Guilherme Oliveira Guerra. -- Caraguatatuba, 2015.

48 f.

Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) -- Instituto Federal de São Paulo, Câmpus Caraguatatuba, 2015.

1. Android (Sistema operacional). 2. Java virtual machine (Programa de computador). 3. Compiladores (Computadores). I. Título.

CDD 005.43

ANÁLISE DO DESEMPENHO DAS TÉCNICAS DE PROGRAMAÇÃO RECOMENDADAS PARA O ANDROID NAS MÁQUINAS VIRTUAIS ART E DALVIK

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Tecnólogo, da Área de Informática, do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo.

Orientador:
Prof. Eduardo Pereira de Sousa

CARAGUATATUBA

2015

TERMO DE APROVAÇÃO

ANÁLISE DO DESEMPENHO DAS TÉCNICAS DE PROGRAMAÇÃO RECOMENDADAS PARA O ANDROID NAS MÁQUINAS VIRTUAIS ART E DALVIK

por

ACÁCIO ALKMIN DE ALMEIDA CASSIANO
GUILHERME OLIVEIRA GUERRA

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 6 (seis) de Julho de 2015 como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas (ADS). Os candidatos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados, a qual após deliberação, considerou o trabalho aprovado.

_____
Prof. Eduardo Pereira de Sousa
Prof. Orientador

_____
Ederson Rafael Wagner Presidente

_____
Luiz Antonio Rodrigues Junior Membro

“O maior inimigo do conhecimento não é a ignorância, mas a ilusão do conhecimento”.

Stephen Hawking

AGRADECIMENTOS

Agradecemos a Deus por ter nos dado saúde e força para superar as dificuldades encontradas para realizar este trabalho de conclusão de curso.

Ao Instituto Federal de Educação, Ciência e Tecnologia Câmpus Caraguatatuba, que nos deu toda a estrutura necessária para chegarmos até aqui. Agradecemos a todo corpo docente, a direção e administração.

Agradecemos ao nosso orientador Eduardo Pereira de Sousa, pelo suporte no pouco tempo que lhe coube, pelas suas correções e incentivos.

Agradecemos ao colega Victor Ulisses Pugliese que contribuiu para o desenvolvimento desse projeto.

E a todos as pessoas que direta ou indiretamente fizeram parte da nossa formação, o meu muito obrigado.

AGRADECIMENTOS - ACÁCIO ALKMIN

Primeiramente a Deus que permitiu que tudo isso acontecesse, ao longo de minha vida, e não somente nestes anos como universitário, mas que em todos os momentos é o maior mestre que alguém pode conhecer.

À Instituição pelo ambiente criativo e amigável que proporciona.

Ao professor Eduardo, pela orientação, apoio e confiança.

Agradeço a minha mãe Eliane, heroína que me deu apoio, incentivo nas horas difíceis, de desânimo e cansaço. E mostrando que vivemos a cada dia em uma intensa batalha e que sim, podemos sair vitoriosos.

Ao meu pai Anderson que apesar de todas as dificuldades me fortaleceue me deu toda estrutura necessaria e que para mim foi muito importante.

Agradeço a minha namorada que mesmo ela sentindo saudades me deu apoio em todo os momentos que eu precisei mesmo estando longe.

Meus agradecimentos aos familiares, amigos, companheiros de trabalhos e irmãos na amizade que fizeram parte da minha formação e que vão continuar presentes em minha vida com certeza. A todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

AGRADECIMENTOS – GUILHERME OLIVEIRA GUERRA

Gostaria de agradecer primeiramente a Deus por me dar força a cada dia para realização desse Curso e não me abandonar em nenhum momento.

Agradeço minha mãe Marta Regina e meu pai Daniel Romero que não desistiram de mim em nenhum momento e sempre depositaram fé que eu poderia concluir esse curso.

Tenho que agradecer o Instituto Federal campus Caraguatatuba por fazer possível essa conquista na minha vida, fornecendo a estrutura que foi necessáriae o professor que nos orientou Eduardo Pereira de Sousa. Agradeço a outros professores que influenciaram nesse trabalho.

Agradeço a todos os meus familiares e amigos que direta ou indiretamente influenciaram no meu curso fazendo parte da minha vida durante esse período de formação. Espero que todos continuem sendo presentes na minha vida e que seja possível vivermos mais experiências juntos.

A minha namorada que mesmo longe de mim sempre me incentivou para concluir esse curso.

RESUMO

O sistema operacional Android é uma plataforma de software livre amplamente utilizada em smartphones, que possibilita o desenvolvimento de aplicativos por meio da linguagem de programação Java. Os aplicativos desenvolvidos para Android até sua versão 4.4.2 são executados pela máquina virtual Dalvik, que realiza a compilação e otimização do código-fonte Java. Esse processo de compilação e execução passou por uma grande mudança a partir da versão 5.0 do sistema operacional, onde a máquina virtual Android RunTime (ART) foi introduzida, substituindo a predecessora Dalvik. Este trabalho visa verificar a pertinência das técnicas de programação originalmente propostas para a máquina virtual Dalvik do Android, diante transição para a ART.

Palavras-chave: Android. Máquinas Virtuais. Compilador. Dalvik. ART.

ABSTRACT

Android operational system is an open-source platform widely used in smartphones, which allows the development of applications written in Java programming language. Applications developed for Android until its 4.4.2 version are run by a virtual machine called Dalvik, which performs the compilation and optimization of the Java source code. This process of compilation and execution underwent a major change from version 5.0 of the operating system where the virtual machine Android RunTime (ART) was introduced, replacing its predecessor Dalvik. This work aims to verify the relevance of programming techniques originally proposed for Android's Dalvik virtual machine given the transition to ART.

Keywords: Android. Virtual Machine. Compilation. Dalvik. ART.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS, SIGLAS E ACRÔNIMOS

LISTA DE ABREVIATURAS

MV	Maquina Virtual

LISTA DE SIGLAS

ART	AndroidRunTime
CPU	Unidade Central de Processamento
OHA	Open Handset Alliance
API	Interface de Programação de Aplicativos
SDK	Software Development Kit
AOT	Aheadof time

SUMÁRIO

1. INTRODUÇÃO

1.1 Contexto da Pesquisa

A execução de aplicativos no sistema operacional Android é baseada no uso de máquinas virtuais. Essas máquinas são responsáveis pela compilação do *bytecode* Java [1] em instruções que serão executadas pelo sistema.

Dado o contexto no qual os aplicativos Android são executados, algumas recomendações em relação a técnicas de programação para criação de aplicativos são feitas pelo próprio desenvolvedor do sistema operacional, a empresa Google. Essas técnicas, chamadas *PerformanceTips,* visam otimizar a *performance* desses aplicativos.

1.2 Objetivo

Este trabalho tem por objetivo avaliar as técnicas de programação originalmente sugeridas para o sistema operacional Android, verificando sua validade no contexto da nova máquina virtual ART. Serão realizados testes de desempenho para as técnicas apresentadas, levando em conta a *performance* perceptível pelo usuário por meio do tempo de execução dos testes, bem como o uso do processador para cada um dos testes.

Os testes serão aplicados em ambientes reais utilizando dispositivos móveis como *smartphones,* e em ambientes virtualizados com a utilizaçãode um emulador de Android. Será feita uma análise comparativaentre as máquinas virtuais com e sem a aplicaçãodas técnicas de programação.

1.3 Organização do Texto

No Capítulo 2 serão apresentados os conceitos referentes aos temas que serão abordados durante o trabalho, como o Android, as técnicas de programação e a linguagemJava. No Capítulo 3 são apresentados os trabalhos utilizados como base para desenvolvimento desse projeto, a delimitação do escopo do trabalho, e o modelo proposto para odesenvolvimento e análise dos dados. No Capítulo 4 é apresentado um estudo de caso onde os testes são aplicados, a forma do seu desenvolvimento, as ferramentas que foram utilizadas e as etapas de aplicação. No Capítulo 5 são apresentados os resultados obtidos com o estudo de caso. No Capítulo 6 são exibidas as conclusões finais, limitações encontradas e trabalhos futuros.

2. CONTEXTUALIZAÇÃO

Neste capitulo será apresentado o sistema operacional Android, as máquinas virtuais que serão objetos de teste, a linguagem de programação utilizada para desenvolvimento das aplicações e as técnicas de programação recomendadas, cuja *performance* será avaliada em cada uma das máquinas virtuais.

2.1 Sistema Operacional Android

Com o incontrolável crescimento do mercado de dispositivos móveis no século XXI, a Google e outras empresas de telefonia móvel viram a oportunidade e a necessidade de apoiar a criação de uma entidade com o foco em telefonia móvel dando origem a *Open Handset Alliance* (OHA) [2].

A OHA é uma aliança com mais de 40 empresas envolvidas, desde operadoras de telefonia celularaté empresas de comercialização de aparelhos. Esta aliança teve por objetivoo desenvolvimento deum sistema operacional de código aberto onde qualquer usuário pudessecriar e instalar aplicativos que tivessem os mesmos privilégios que os aplicativos de fabricantes têm, tornando o dispositivo cada vez mais livre e interessante [3].

O primeiro projeto envolvendo todas as áreas da OHA foi nomeado Android, um sistema operacional completo baseado no Linux, para dispositivos móveis possuindo *interface* para o usuário, aplicativos e *middleware*[4].

O grande objetivo por trás do Android é permitir que a comunidade possa desenvolver aplicações, usufruindo de toda a capacidade do dispositivo móvel e assim contribuir com a evolução do sistema a cada dia. O Android é o primeiro sistema para dispositivos móveis com código aberto e, além disso, a OHA disponibiliza o Android SDK que é umkit para o desenvolvimento de

aplicaçõesquepossui a API e as ferramentas necessárias para o desenvolvimento na linguagem JAVA [5].

2.2 Máquinas Virtuais do Android

O Androidutiliza desde a sua primeira versão a Máquina Virtual (MV) Dalvik para execuçãodas aplicações desenvolvidas em Java. A partir da versão 4.4 (Android*KitKat*), uma nova MV foi disponibilizada. Esta versão, conhecida como *AndroidRunTime* (ART), foi criada com o intuito de melhorar o desempenho na execução das aplicações, otimizando a utilização dos recursos de *hardware* existentes nos *smartphones* modernos [6].

2.3 Máquina Virtual Dalvik

A Máquina virtual Dalvik foi projetada e escrita por Dan Bornstein com contribuições de outros engenheiros da Google como parte da plataforma Android para telefones celulares.

É uma máquina virtual otimizada para fazer bom uso dos recursos limitados disponíveis nos *smartphones* na época de ser desenvolvimento e projetada para ser executada em várias instâncias simultâneas, deixando a cargo do sistema operacional o isolamento de processos, gerenciamento de memória e suporte a processamento paralelo [7].

Sua função é compilar o *bytecode* Java contido nos arquivos DEX (*DalvikExecutable*) [8]das aplicações Android, convertendo-os em código executável pelo *Smartphone*. A MV Dalvik tem por característica a compilação desses arquivos sob demanda, ou seja, no momento em que a aplicação é executada.

2.4 Máquina Virtual ART

A Máquina Virtual *AndroidRunTime* (ART) tem por objetivo substituir a máquina virtual Dalvik.Seu principal diferencial é a compilação do *bytecode*das aplicaçõesdurante a instalação das mesmas, fazendo maior uso da memória, mas com o benefício de uma execução mais eficiente dos aplicativos, com melhor *performance*.

A Máquina Virtual ART, deve ser capaz de compilar todos os arquivos DEX originalmente gerados para a MV Dalviksem dificuldades.O formato de arquivo DEX foi desenvolvido especificamente para atender as necessidades do sistema operacional Android [2].

2.3 Linguagem de Programação Java

A linguagem de programação Java foi criada em 1995, por funcionários da empresa *Sun* devido a demanda do mercado, que necessitava de uma linguagem versátil e que fosse compatível com diversas plataformas e escalável para o desenvolvimento de grandes projetos [9]. Logo após o seu lançamento, o uso difundiu-se no desenvolvimento de pequenos programas executados em navegadores da *web* e que adicionam interatividade a sites estáticos. Este uso contribuiu imensamente para sua popularização [10].

2.4 Android Studio

O Android Studio proporciona um ambiente de desenvolvimento adequado por permitir a criação de projetos facilitada. A partir disso, o desenvolvimento do código é facilitado com sugestões da própria ferramenta (auto completar e complementos). Durante a criação, caso existam erros de sintaxe os mesmos são destacados pela ferramenta para correção. Após a finalização é possível a execução no ambiente de teste simulado ou real,por meio do qual é possível a verificação da eficácia do projeto criado,caso existam erros a identificação é exibida de forma simples,

apontando no console onde eles se encontram; não havendo erros o Aplicativo é executado no ambiente de teste e os resultados são exibidos.

2.5 Técnicas de Programação

As técnicas de programação que serão utilizadas no projeto e que são sugeridas como boas práticas de programação são:

- Evitar *Gettere**Setter*.
- Utilizar Métodos Estáticos.
- Uso da sintaxe aprimorada para aestrutura de repetição *For*
- Evitar o uso de ponto flutuante.

O site oficial voltado para desenvolvedores de aplicações Android recomenda ainda outras técnicas, porém para este trabalho optou-se pelo teste apenas das técnicas listadas acima, por compreender que o conjunto de técnicas escolhido abrange a maior parte dos projetos de aplicações para Android [11].

2.5.1 Evitar GettereSetter

Em linguagens orientadas a objetoé uma prática comum o uso de *Gettere**Setter* interno ao invés de acessar o campo diretamente, pois o compilador geralmente pode sequenciar o acesso a esses campos, otimizando-o [12].

O uso de*Getter* e *Setter*é mais custoso em termos de processamento do que acesso direto as propriedades, pois estes métodos não necessariamente devem ter uma propriedade correspondente, ou seja, não há uma relação 1 para 1 entre estes métodos e as propriedades da classe. Um método *Getter* pode obter seu valor a partir de diversos campos, assim como uma única chamada a um método *Setter* pode alterar variados campos, assim as chamadas ao método*Getter* e *Setter* não

podem ser otimizadas pelo compilador ao ponto de terem o mesmo desempenho de um acesso direto as propriedades.

Sem um JIT (*Just-in-Time*) que é o caso da Máquina Virtual Dalvik, o acesso direto ao campo é cerca de 3 vezes mais rápido do que uma chamada a um método *Getter*,já com um JITo acesso direto ao campotorna-se 7 vezes mais rápido [11].

2.5.2 Utilização de MétodosEstáticos.

Caso um método de uma classe não necessite de acesso as propriedades dessa classe, recomenda-se que esse método seja declarado como estático, dessa forma é possível obter uma *performance* de 15 a 20% superior do que com o uso de um método virtual convencional. Isso também permite ao desenvolvedor concluir a partir da assinatura do método que as chamada a ele não alteram o estado da classe [11].

2.5.3Uso dasintaxe aprimorada para aestrutura de repetição For

Conhecido também como “*for-each*”, pode ser usado para as coleções que implementam a interface *Iterable* e para vetores.Este tipo de *loop* permite que o compilador otimize a execução do *loop*, através do *cache* do tamanho total do vetor acessado entre outros artifícios [13].

2.5.4 Evitar o uso de ponto flutuante

O uso de tipos com ponto flutuante (*double* e *float*) é em geral 2 vezes mais lento do que o uso de variáveis do tipo inteiro em dispositivos Android. Dessa forma, o uso desses tipos deve se restringir a onde houver real necessidade da precisão numérica oferecida por eles, nas demais situações deve-se usar preferencialmente o tipo inteiro. Cabe ressaltar que não existem diferenças significativas na *performance* dos tipos *double* e *float*, apenas no espaço de armazenamento requerido [11].

3. DESENVOLVIMENTO

Neste capitulo serão apresentados os trabalhos que foram estudados inicialmente para levantar a necessidade da pesquisa realizada por este trabalho. As formas que foram trabalhadas as técnicas de programação nos trabalhos relacionados como referência e como será abordado pelo desenvolvimento deste trabalho.

3.1 Trabalhos Relacionados

Para o desenvolvimento deste trabalho foram levantadas outras iniciativas que contemplassem as técnicas recomendadas para o desenvolvimento na plataforma Android bem como demais boas práticas relacionadas a essa plataforma.

Em busca nas bases de dados disponíveis, foram encontrados dois trabalhos brasileiros que se encaixam no assunto proposto, são eles:

- Análise De Impacto Do Uso De Técnicas De Programação No Desempenho De Aplicações Android [14].
- Avaliação Das Boas Práticas Android Para Desenvolvimento [15].

Ambos os estudos abordam um subconjunto das técnicas de programação recomendadas para a plataforma Android que serão apresentadas neste trabalho. Nestes estudossão apresentados testes de *performance* das recomendações "Evita o uso de *Getter* e *Setter*" e "Uso da sintaxe *For* aprimorada", porém estes limitam-se a analisar o desempenho dessas técnicas somente MV Dalvik, que era a única disponível no momento de seu desenvolvimento.

3.2 Delimitação

Este trabalho se concentrará na análise do impacto de *performance* das técnicas de programação já apresentadas nas máquinas virtuais Dalvik e ART, analisando os resultados do uso e não uso de cada uma dessas técnicas em ambas as MVs. O impacto de *performance* que se deseja mensurar é aquele perceptível pelo usuário, ou seja, o impacto no tempo de resposta das aplicações que façam ou não uso intensivo dessas técnicas. Além disso, a *performance*perceptível pode ser analisada quanto a estabilidade da aplicação e do sistema operacional como um todo, por isso será verificado também o impacto desses teste na utilização do processador.

3.2.1 Técnicas identificadas para o estudo

As técnicas de programação foram selecionadas para este estudo, pois são utilizadas em situações comumente encontradas no desenvolvimento de aplicações para Android e na linguagem Java. Outro aspecto importante dessas técnicas é que seu resultado pode ser facilmente quantificado em relação aos diversos cenários de teste, enquanto algumas das técnicas que não foram selecionadas para este estudo envolvem aspectos qualitativos e subjetivos para sua aplicação e mensuração.

3.2.2 Escolha do objeto de estudo

A análise comparativa das técnicas de programação nas máquinas virtuais Dalvik e ART foi escolhida como objeto de estudo, pois como veremos a seguir já foi demonstrado pelos trabalhos relacionados o seu impacto na *performance* quanto ao uso na máquina virtual Dalvik. Por se tratar o ART de uma máquina virtual ainda não devidamente testada, cujo método de compilação difere totalmente do Dalvik, é importante que se verifique se essas técnicas continuam válidas para essa nova máquina virtual. A seguir exibimos alguns dos resultados obtidos pelos trabalhos relacionados.

3.2.2.1Resultados obtidos pelos trabalhos relacionados [14] e [15].

No trabalho "Avaliação Das Boas Práticas Android Para Desenvolvimento" [15], os testes do uso ou não das técnicas de programação foi repetido 30 vezes para cada instância de teste, e a partir dos tempos de execução obtidos chegou-se as médias apresentadas nas figuras [1] e figura [2] para a aplicação e não aplicação de cada técnica.

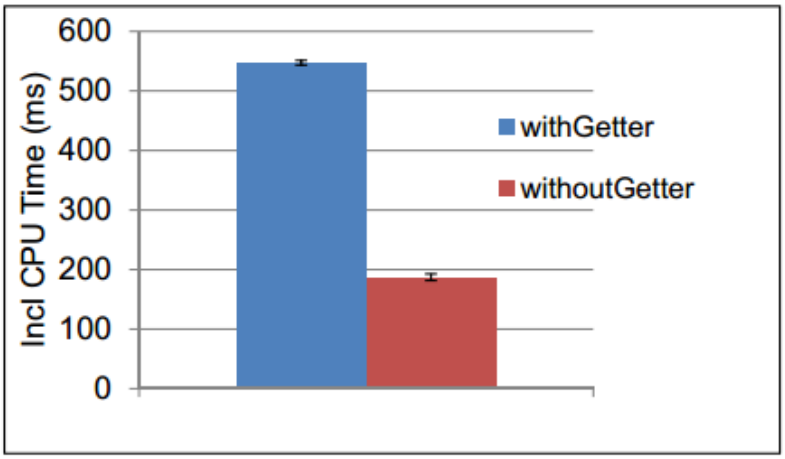

Figura 1 - Resultados da aplicação ou não da técnica “Evitar o uso de*Gettere Setter*”.

Fonte: Tonini, 2013, p 14.

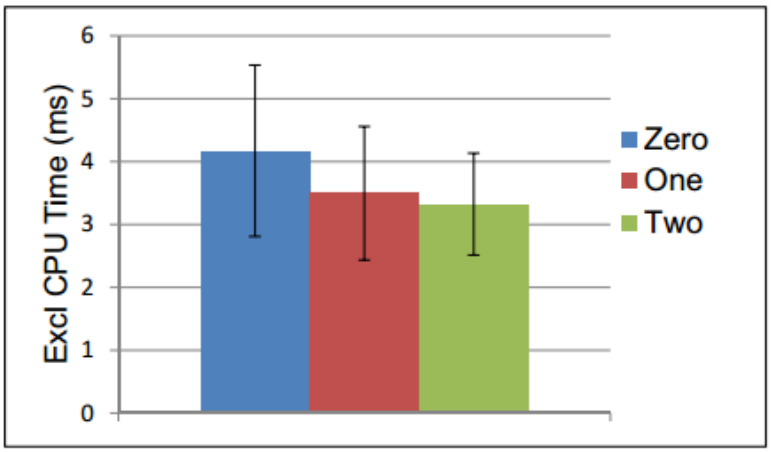

Figura 2 - Resultados da aplicação ou não da técnica “Uso de sintaxe aprimorada para a estrutura de repetição *For*”.

Fonte: Tonini, 2013, p 14.

No trabalho "Análise De Impacto Do Uso De Técnicas De Programação No Desempenho De Aplicações Android" [14] foi realizado o teste em relação ao uso da técnica "Uso da sintaxe *For* aprimorada", os resultados obtidos por meio da ferramenta *Little-Eye* mostraram um uso 22,7% da CPU sem a aplicação da técnica e de 23,5% quando a técnica é aplicada, este trabalho não levou em consideração o tempo de execução dos testes.

3.2 Modelo Proposto

A comparação de desempenho das máquinas virtuais do Android, ART e Dalvik são o foco do projeto, poissão elas as responsáveisporcompilar o*bytecode* dos aplicativos desenvolvidos em Java e transformá-los em código executável por meio do sistema operacional Android. A figura a seguir ilustra a forma de compilação das aplicaçõesAndroiddesenvolvidas em Java.

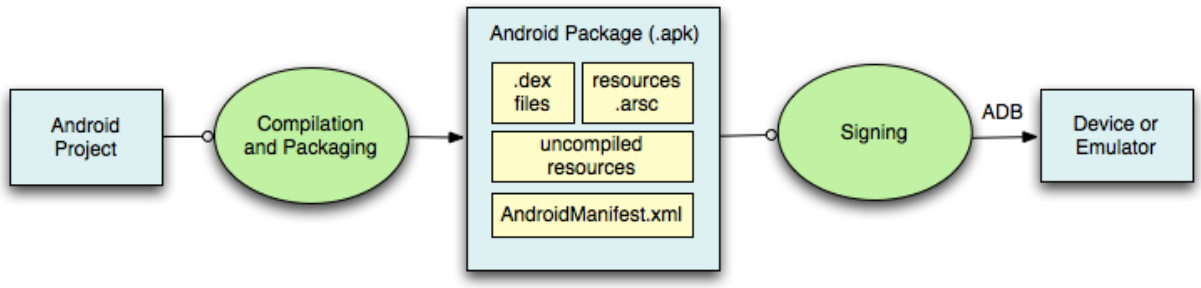

Figura 3 - Processo de compilação aplicações desenvolvidas em Java.

Fonte: *Android Development Guide (ten) establish process*, 2014.

O desenvolvimento a ser apresentado, deve ser feito de forma que sejam testadas as máquinas virtuais do Android para verificar mudanças no desempenho com o uso de técnicas de programação, tendo como base os trabalhos já mencionados. Utilizando como principio apresentar a diferenciação entre as máquinas virtuais, foram escolhidas algumas técnicas de programaçãopara relatar os ganhos das

técnicas e se há algum impacto no uso das máquinas virtuais em relação a essas técnicas.

Com todo o levantamento feito para modelar este trabalho, pode haver um impacto no uso das técnicas de programação, pois há diferenças entre as Máquinas Virtuais do Android com suas formas de compilação. A MV Dalvik, utilizaaexecução *Just-in-Time*(JIT) e a MV ART funciona*Ahead-of-Time* (AOT).

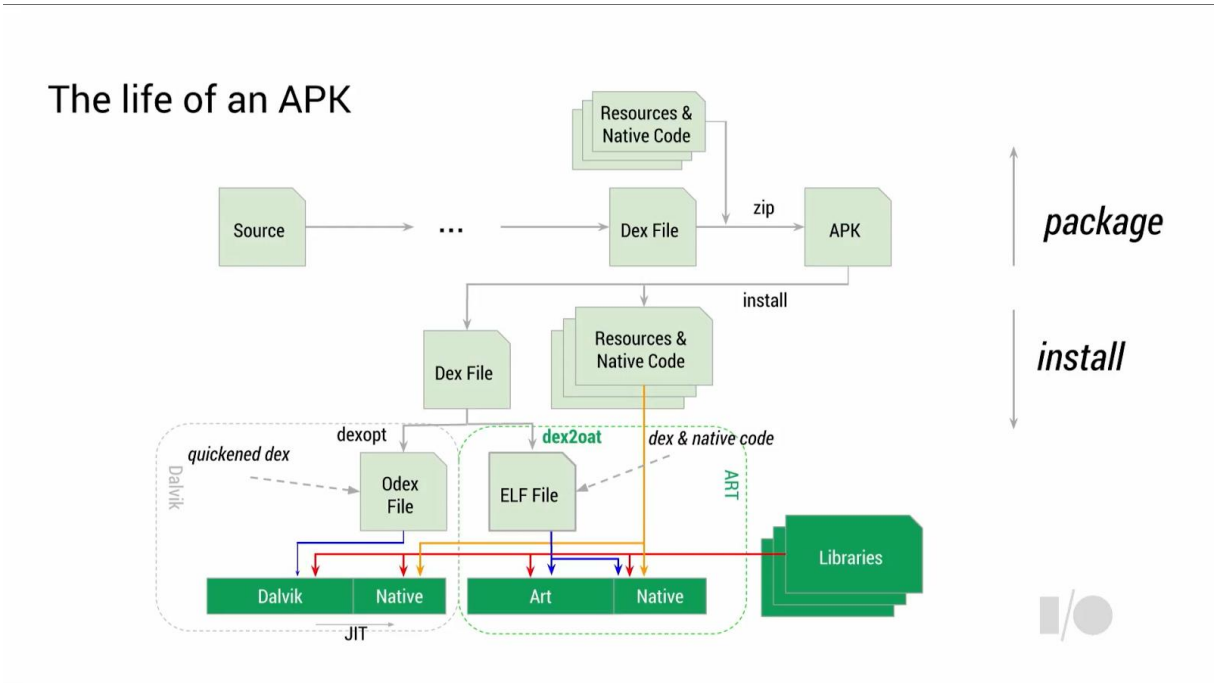

Figura4 – Compilação ART e DALVIK.
Fonte: A Closer Look at Android RunTime (ART) in Android L, 2014.

Como mostra a imagem, o desenvolvimento da aplicação é se dá da mesma forma, porém o resultado final é tratado de forma diferente.

Para êxito deste projeto, é necessário que seja realizado da seguinte forma:

- Desenvolvimento: Criar os aplicativos em Java utilizando as técnicas de programação de forma que realmente estejam utilizando as MV do Android e testando suas diferenças.

- Aplicação: Executar todos os testes criados, executando-os individualmente em ambas as MV.
- Análise: Verificar os resultados da execução de cada um dos testes e analisar para cada um deles os fatores abaixo:
    - Desempenho: Tempo de resposta para execução do aplicativo
    - Estabilidade: Se houveram alterações como aumento anormal nautilização da CPU, lentidão, mal funcionamento ou se manteve estável.
- Resultados: criar relatórios, gráficos e visibilidades para melhor entendimento do trabalho realizado.

4.ESTUDO DE CASO

Neste capítulo, será apresentado o projeto onde foi aplicada a análise do desempenho das técnicas de programação recomendadas para o Android nas máquinas virtuais ART e Dalvik, com o intuito de relatar os testes desenvolvidos em aplicações práticas, mostrando sua viabilidade de desenvolvimento e pertinência ao objeto de estudo.

4.1 Introdução ao Estudo de Caso

Para aplicação dos testes foi necessário inicialmente conhecer o programa Android Studio [16], utilizado no desenvolvimento de aplicações para Android por meio da linguagem Java.Também foi necessário o acesso a ambientes de simulação (*smartphones* virtuais) e ambientes de teste reais (*smartphones*) para execução da aplicação desenvolvida para realização dos testes e obtenção dos resultados.

4.2 Requisitos utilizados

Levantadas as necessidades, foi decidido quais seriam as tecnologias necessárias para a realização do projeto, que foram definidas conforme a **tabela 1**:

Tabela 1- Tecnologias para desenvolvimento dos códigos.

Tecnologia para Desenvolvimento dos códigos	
Android Studio	1.2
Java Runtime Enviroment - JRE	1.7.0_79-b15
Java Development Kit - JDK	7.0.790.15
Sistema Operacional para Desenvolvimento	Windows 8.1

A figura abaixo mostra o ambiente de teste disponibilizado pelo Android Studio, onde foram realizados alguns testes deste projeto.

Figura5 – Ambiente de teste*Android Studio*.

4.4 Criação dos códigos

Para a realização dos testes e obtenção dos resultados, foi necessário o desenvolvimento do código-fonte para cada um dos testes citados nesse trabalho, os quais foram separados em classes individuais.

4.4.1 Classes

Cada um dos testes foi criado a partir de uma classe Java, utilizada para individualizar os testes e seus resultados.

4.4.1.1. Classe *Getter* e classe *Setter*

No primeiro teste foi criada uma classe com o intuito de realizar o teste com *Gettere Setter* e sem o *Gettere Setter*.

A figura 6 mostra como foi criado o código da classe *Getter*, como foi aplicado o desenvolvimento do código para realização do teste.

```
public class Getter {
    private int getter = 10000000;

    int getGetter(){
        return getter;
    }

    double withoutGetter(){
        return getter/777777777;
    }

    double withGetter(){
        int i;
        i = getGetter();
        return i/777777777;
    }
}
```

Figura6 – Código com *Getter* e *Setter* e sem *Getter* e *Setter*.

4.4.1.2. Classe Métodos Estáticos

No segundo teste foi criada uma classe com o intuito de realizar o teste com Métodos Estáticos e sem Métodos Estáticos.

A figura 7 mostra como foi criado o código da classe com método estático e sem método estático.

```
public class Variavel {
    static int somar2numeros_static(int num1, int num2){
        return num1 + num2;
    }
    int somar2numeros(int num1, int num2) { return num1 + num2; }
}
```

Figura7 – Código com Método Estático e sem Método Estático.

4.4.1.3. Classe DoFor (*For* aprimorada)

No terceiro teste foi criada uma classe com o intuito de realizar o teste com sintaxe *For* aprimorada e sem a sintaxe aprimorada.

A figura 8 mostra como foi criado o código da classe sem *FOR* aprimorado e com *FOR-EACH*.

```java
public class DoFor {
    public void zero(){
        int sum = 0;
        int[] vetor = new int[1000];
        for(int i = 0; i < vetor.length; i++){
            vetor[i] = i+1;
        }
        for(int i = 0; i < vetor.length; i++){
            sum += vetor[i];
        }
    }
    public void two(){
        int sum = 0;
        int[] vetor = new int[]{0,1,2,3,4,5,6,7,8,9,10};
        int i=0;
        for(int v : vetor) {
            sum += v;
            i++;
        }
    }
```

Figura8 – Código da classe sem *For* aprimorado e com *For-Each*.

4.4.1.4. Classe *Float*Inteiro

No quarto teste foi criada uma classe com o intuito de realizar o teste com a multiplicação de números inteiros e a soma de números *float*.

A figura 9 mostra como foi criado o código da classe fazendo a multiplicação de números *float*e a multiplicação de números inteiros.

```
public class FloatInteiro {

    float x = 0;
    int y = 0;

    int MultiplicarInteiro(){

        return 48485548*48485548;
    }

    float MultiplicarFloat(){

        return 48485548*48485548;
    }

}
```

Figura9 – Código da classe Multiplicação de números *float*e a Multiplicação de números inteiros.

4.5Ambientes de teste

Os ambientes definidos para obter os resultados necessários foram:

Tabela 2 –Ambientesde testes (*Smartphones* e MV Android Studio).

Modelo	Versão S.O Android	Tipo
LG-D855P	4.4.2	Ambiente de teste Real
Razr i XT890	4.4.2	Ambiente de teste Real
Nexus 5	4.4.2	Ambiente de teste Virtual
Nexus 4	4.4.2	Ambiente de teste Virtual

Foram escolhidas todas as máquinas de teste, com a mesma versão do Android, pois a versão 4.4.2 possui como opção para desenvolvedoresa possibilidade de alternar entre as MV Dalvik e ART.

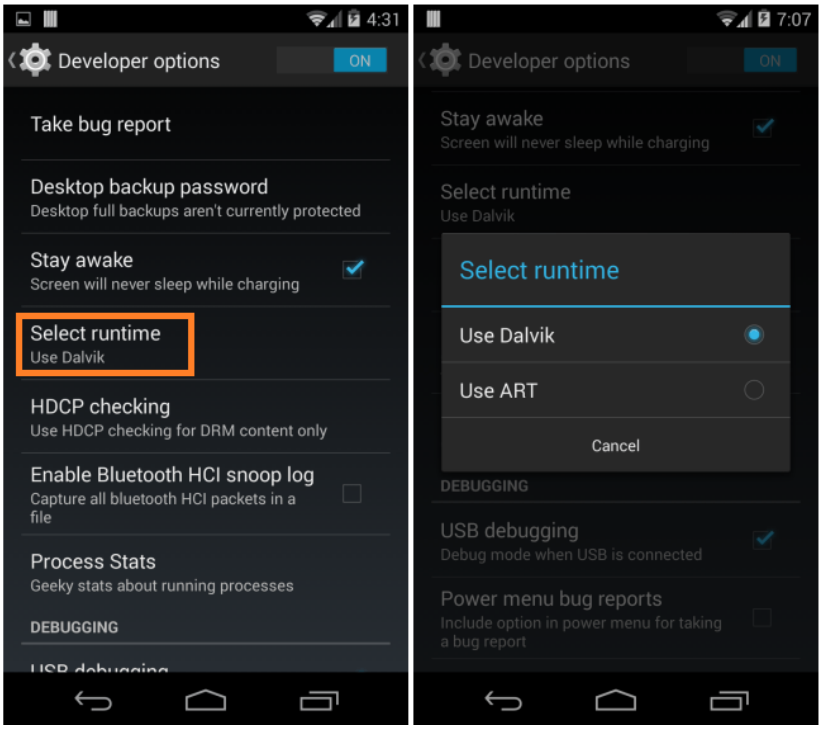

Figura 10 – Seleção da maquina virtual ART ou Dalvik.

4.6 Descrição das métricas do projeto

Para realizar a criação dos gráficos de resultados do projeto, foram estabelecidas métricas para aplicação. Foi necessário realizar cada teste desenvolvido nos dispositivos e maquinas virtuais escolhidas, onde dentro da aplicação é cada teste é repetido por 50 vezes, o tempo de início e encerramento de cada teste é registrado para o calculo do tempo de execução. Para os testes de CPU foram realizadas 5 repetições a partir das quais foi obtida a utilização média de CPU de cada teste.

4.6.1 Etapas aplicadas

A seguir são apresentados os passos que foram utilizados para obter os resultados:

- Passo 1: Escolher o teste que será aplicado, o smartphone e a máquina virtual.
- Passo 2: Aguardar a execução do aplicativo de teste.

- Passo 3: Coletar os dados de tempo de execução do aplicativo e o uso de CPU do dispositivo.
- Passo 4: Armazenar os dados em tabelas.
- Passo 5: Fechar o aplicativo e mudar as configurações para a próxima máquina virtual que será testada.
- Passo 6: Executar o teste novamente e assim sucessivamente com todos os testes, *smartphones* e máquinas virtuais.

4.7 Conclusão do Estudo de Caso

Os dados levantados nesse capitulo foram necessários para que fosse possível obter os resultados dos testes e realizar a análise comparativa entre as MVDalvik e ART com e sem a aplicação das técnicas de programação anteriormente mencionadas.

5 RESULTADOS OBTIDOS

Neste capítulo serão apresentados os resultados dos testes realizados para comparação das técnicas de programação e das máquinas virtuais do Android.

5.1 Resultados em Tempo de execução do aplicativo

Os testes realizados com o objetivo de relatar o tempo de execução entre as máquinas virtuais utilizando técnicas de programação não apresentaram grandes diferenças em todas as avaliações realizadas. Abaixo é apresentada uma tabela com o resultado médio das execuções de cada um dos testes nos diversos ambientes de teste utilizados.

As informações da tabela mostram os resultados dos testes realizados duzentas vezes, cinquenta vezes por *smartphone*.

Tabela 3 - Resultados dos testes em tempo de execução.

DALVIK		ART	
TESTE REALIZADO	TEMPO DE EXECUÇÃO	TESTE REALIZADO	TEMPO DE EXECUÇÃO
Sem getter e setter	< 0.10	Sem getter e setter	0.30
Com getter e setter	0.10	Com getter e setter	0.30
Sem for each	0.70	Sem for each	0.30
Com for each	0.10	Com for each	0.20
Com estatico	< 0.10	Com estatico	0.40
Sem estatico	0.10	Sem estatico	0.20
Utilizando int	0.10	Utilizando int	0.10
Utilizando float	0.50	Utilizando float	0.20

5.2 Resultados *Performance*da CPU

Ao longo da realização dos testes foram percebidas diferenças significativas no uso da CPU ao executar alguns dos testes. Dessa forma, foram armazenados os dados para serem analisados e comparados

5.3 Resultados dos obtidos por técnica de programação

5.3.1 Resultados de “Evitar *Getter e Setter*”

No primeiro gráfico dos resultados obtidos referentes a performace de CPU dos testes executados nesse projeto, será apresentado os resultados dos testes realizados com a técnica de programação "Evitar *Getter* e *Setter*", onde serão apresentadosos resultados de uma forma geral, relatando uma média entre todos os smartphones utilizados na realização dos testes.

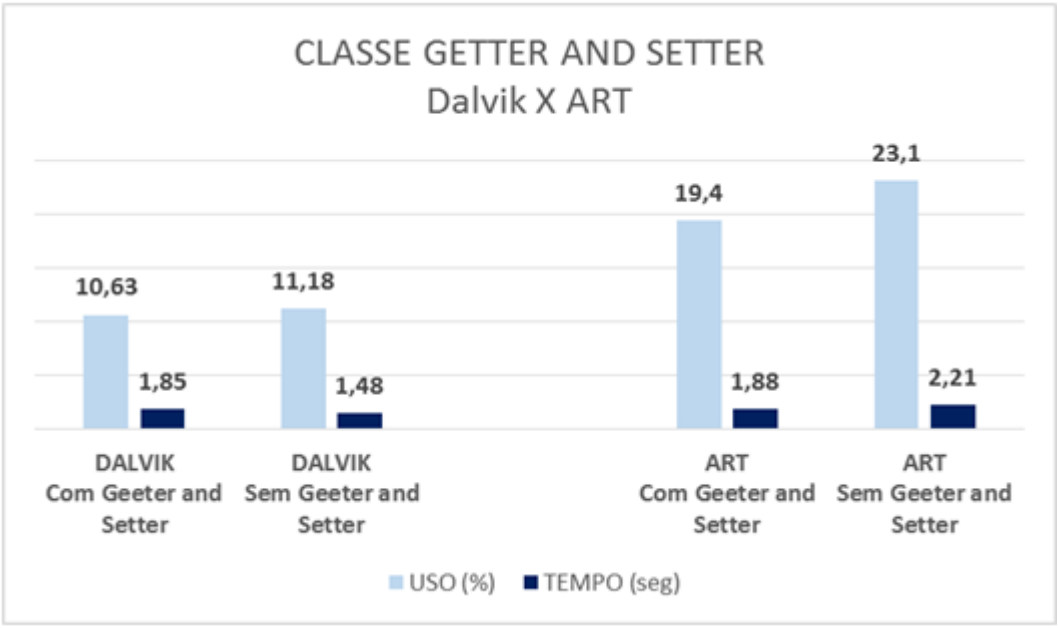

Figura 11- Gráfico dos resultados de CPU “*Getter* e *Setter”*.

Na tabela a seguir é apresentado os resultados individuais de cada um dos smartphones que foram utilizados para o teste "*Getter* e *Setter*". Onde são apresentados em porcentagem e em tempo de execução. Onde, pode-se notar grandes diferenças entre os resultados dos smartphones.

Tabela 4 - Resultados dos testes “Evitar *GettereSetter”* individuais por *smartphone*.

CLASSE GETTER AND SETTER - ART		
Com Getter and Setter		
CPU	USO	TEMPO
Nexus 5	33%	2,3 seg
Nexus 4	36%	1,6 seg
LG-D855P	1,6%	0,6 seg
Razr I XT890	7%	3 seg

CLASSE GETTER AND SETTER - DALVIK		
Com Getter and Setter		
CPU	USO	TEMPO
Nexus 5	16,7%	2 seg
Nexus 4	18,6%	2,3 seg
LG-D855P	0,7%	0,6 seg
Razr I XT89	6,5%	2,5 seg

Sem Getter and Setter		
CPU	USO	TEMPO
Nexus 5	56%	3 seg
Nexus 4	28,30%	3 seg
LG-D855P	1,6%	0,83 seg
Razr I XT890	6,5%	2 seg

Sem Getter and Setter		
CPU	USO	TEMPO
Nexus 5	11,3%	1,3 seg
Nexus 4	26,6%	2 seg
LG-D855P	0,8%	0,6 seg
Razr I XT89	6%	2 seg

5.3.2 Resultados de “Utilização de Métodos Estáticos”

Neste gráfico será apresentado os resultados dos testes realizados com a técnica de programação "Utilização de Métodos Estáticos", onde serão apresentadosos resultados de uma forma geral, relatando uma média entre todos os smartphones utilizados na realização dos testes. Os números apresentados são as médias das somas individuais de cada dispositivo.

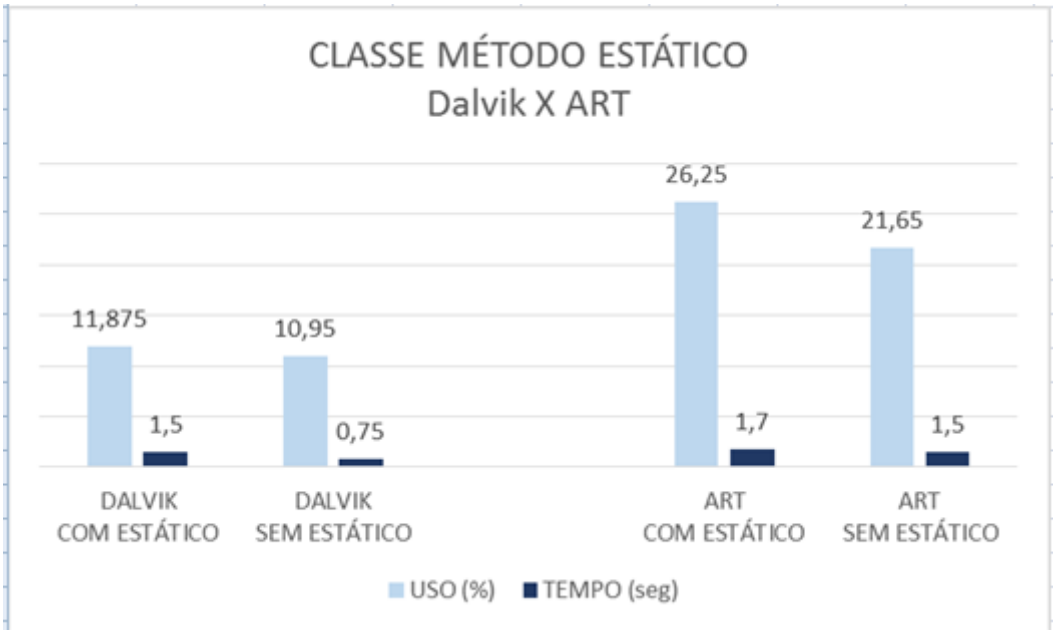

Figura 12- Gráfico dos resultados de CPU “Métodos Estáticos”.

Na tabela a seguir é apresentado os resultados individuais de cada um dos smartphones que foram utilizados para o teste "Utilização de Métodos Estáticos". Onde são apresentados em porcentagem e em tempo de execução.

Tabela 5 - Resultado dos testes dos “métodos estáticos” individuais por *smartphone*.

VARIAVEL ESTÁTICA - ART		
Com estática		
CPU	USO	TEMPO
Nexus 5	42,3%	2,3 seg
Nexus 4	55%	2 seg
LG-D855P	1,2%	1 seg
Razr I XT890	6,5%	1,5 seg
Sem estática		
CPU	USO	TEMPO
Nexus 5	34,6%	2 seg
Nexus 4	44%	2 seg
LG-D855P	1%	1 seg
Razr I XT890	7%	1 seg

VARIAVEL ESTÁTICA - DALVIK		
Com estática		
CPU	USO	TEMPO
Nexus 5	18,3%	2 seg
Nexus 4	23,0%	2 seg
LG-D855P	0,7%	1 seg
Razr I XT89(	5,5%	1 seg
Sem estática		
CPU	USO	TEMPO
Nexus 5	13,7%	2,3 seg
Nexus 4	23,7%	1,6 seg
LG-D855P	0,4%	1 seg
Razr I XT89(	5,5%	1 seg

5.3.3 Resultados de “Uso da sintaxe aprimorada para a estrutura de repetição *For*”

No gráfico será apresentado os resultados dos testes realizados com a técnica de programação " Uso da sintaxe aprimorada da estrutura de repetição *For* ". Os números apresentados são as médias das somas individuais de cada dispositivo, onde são notadas as diferenças entre as máquinas virtuais ART e Dalvik.

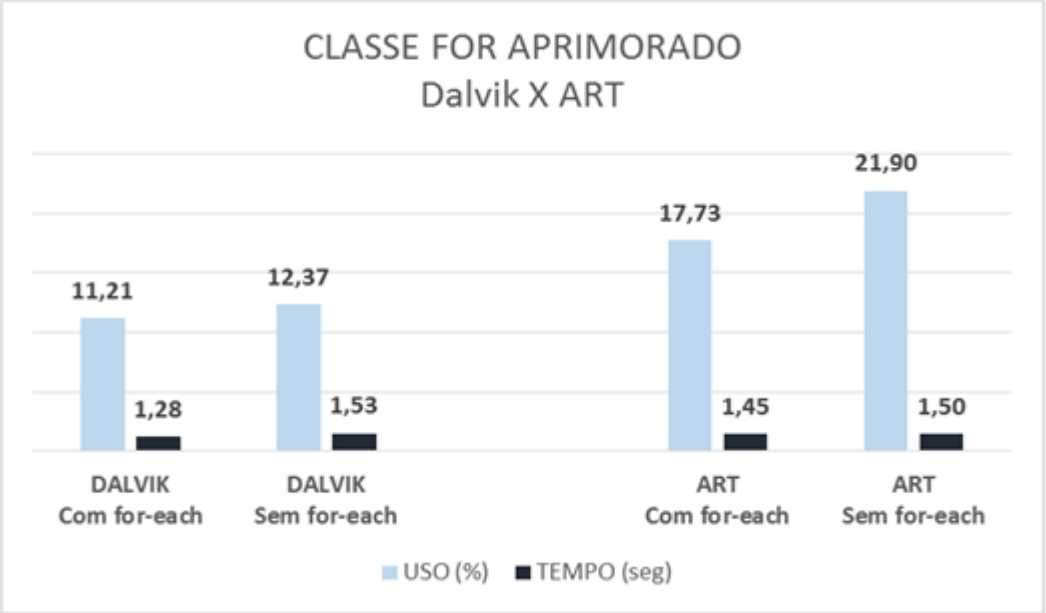

Figura 13- Gráfico dos resultados de CPU “Sintaxe aprimorada da estrutura de repetição *for”.*

Na tabela a seguir é apresentado os resultados individuais de cada um dos smartphones que foram utilizados para o teste "Uso da sintaxe aprimorada da estrutura de repetição *For*". Onde são apresentados em porcentagem e em tempo de execução.

Tabela 6 - Resultado dos testes “utilizar sintaxe aprimorada da estrutura de repetição *for”*.

FOR - ART		
Com for-each		
CPU	USO	TEMPO
Nexus 5	30,7%	2 seg
Nexus 4	32,6%	1,3 seg
LG-D855P	1,6%	1 seg
Razr I XT890	6%	1,45 seg

FOR APRIMORADO - DALVIK		
Com for-each		
CPU	USO	TEMPO
Nexus 5	13%	2,3 seg
Nexus 4	26%	1,3 seg
LG-D855P	0,83%	0,5 seg
Razr I XT890	5%	1 seg

Sem for-each		
CPU	USO	TEMPO
Nexus 5	32,7%	2,3 seg
Nexus 4	47,6%	1,6 seg
LG-D855P	1,3%	1,5 seg
Razr I XT890	6%	1,5 seg

Sem for-each		
CPU	USO	TEMPO
Nexus 5	14,30%	2 seg
Nexus 4	28%	1,6 seg
LG-D855P	0,67%	1 seg
Razr I XT890	6,5%	1,5 seg

5.3.4 Resultados de “Evitar o uso de ponto flutuante”

No último gráfico, será apresentado os resultados dos testes realizados com a técnica de programação "Evitar o uso de ponto flutuante", onde será apresentado o consumo de CPU maior da Máquina Virtual ART em relação a Máquina Virtual Dalvik. Os números apresentados são as médias das somas individuais de cada dispositivo.

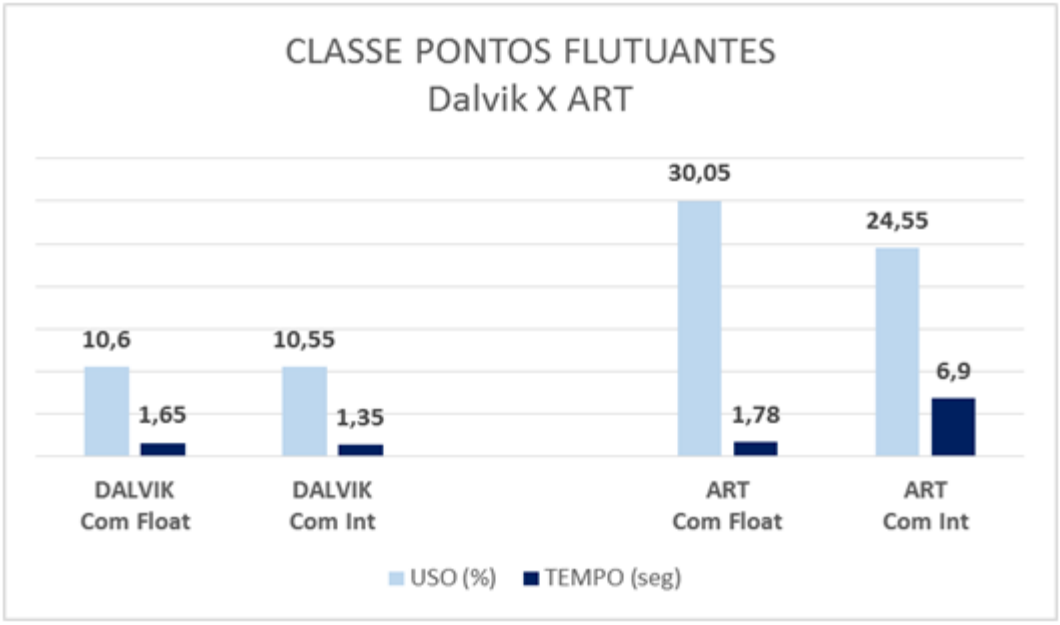

Figura 14 –Gráfico dos resultados de CPU“Evitar o uso de pontos flutuantes”.

Na tabela a seguir é apresentado os resultados individuais de cada um dos smartphones que foram utilizados para o teste "Evitar o uso de pontos flutuantes". Onde são mostrados em porcentagem e em tempo de execução, onde, pode-se notar grandes diferenças entre os resultados dos smartphones. O Nexus 5 chega a utilizar 58,3% da CPU e o LG-D855P usa 1,30% no uso de float com ART, já no Dalvik com o Nexus 5 utiliza 13,3% e o LG-D855P usa 1,3%.

Tabela 7 - Resultado dos testes“Evitar uso de pontos flutuantes” individuais por *smartphone*.

CLASSE PONTOS FLUTUANTES - ART		
Com Float		
CPU	USO	TEMPO
Nexus 5	58,3%	1,6 seg
Nexus 4	51,6%	2 seg
LG-D855P	1,30%	1,5 seg
Razr I XT890	9%	2 seg
Com Int		
CPU	USO	TEMPO
Nexus 5	40,3%	2,6 seg
Nexus 4	50,6%	2,3 seg
LG-D855P	1,30%	1 seg
Razr I XT890	6%	1 seg

CLASSE PONTOS FLUTUANTES - DALVIK		
Com Float		
CPU	USO	TEMPO
Nexus 5	13,3%	1,3 seg
Nexus 4	21,3%	2 seg
LG-D855P	1,3%	1,3 seg
Razr I XT890	6,5%	2 seg
Com Int		
CPU	USO	TEMPO
Nexus 5	13,0%	1,3 seg
Nexus 4	22,7%	1,6 seg
LG-D855P	1%	1 seg
Razr I XT890	5,5%	1,5 seg

6. CONCLUSÃO

Este capítulo mostra as conclusões finais, contribuições, limitações e trabalhos futuros.

6.1 ATENDIMENTO AOS OBJETIVOS PROPOSTOS

O trabalho se propôs a realizar uma comparação entre as técnicas de programação sugeridas no desenvolvimento de aplicações para Android entre a máquina virtual originalmente utilizada pelo Android, Dalvik e a nova máquina virtualART. Estes testes tiveram por objetivo analisar o desempenho percebido pelo usuário e possíveis impactos no uso do processador, avaliou-se a validade das técnicas originalmente sugeridas para o Dalvik e no ambiente mais recente, o ART.

Durante os testes foi verificado que o desempenho percebido pelo usuário é similar entre as duas máquinas virtuais, com pequenas variações a favor de uma ou outra em determinados cenários de teste. Foi notado um maior uso do processador na execução dos testes na máquina ART, com ou sem o uso das técnicas de programação recomendadas. Nos testes de uso da CPU verificou-se que apesar das diferenças apresentadas, a razão entre os valores obtidos para a aplicação ou não das técnicas de programação manteve-se similar.

Conclui-se que as técnicas recomendadas continuam válidas no uso da MV ART, as diferenças de desempenho em relação ao tempo de execução são bastante sutis para serem percebidas pelo usuário, enquanto as diferenças em processamento apresentaram-se em todos os testes, com ou sem o uso das técnicas recomendadas, não sendo decorrentes do uso destas.

6.2 Contribuições

Neste trabalho foram apresentadas comparações entre as Máquinas Virtuais do Android utilizando técnicas de programação, visando contribuir com a comunidade de desenvolvimento para Android diferenciando e relatando os benefícios e os malefícios de cada uma delas.

6.3 LIMITAÇÕES

Durante o trabalho percebeu-se diferenças demasiadamente sutis na *performance* dos testes, uma análise mais aprofundada dessas diferenças requer a utilização de ferramentas específicas para extração de dados mais detalhados e de maior precisão, bem como uma análise mais abrangente do código fonte do próprio sistema operacional Android e dos algoritmos de otimização utilizados pelaMV.

6.4 TRABALHOS FUTUROS

Nos testes verificou-se diferenças significativas de processamento entre os testes realizados na máquina Dalvik e ART. Tais diferenças necessitam de uma análise aprofundada que está além do escopo deste trabalho.

Além das técnicas analisadas neste trabalho existem outras técnicas recomendadas para a plataforma Android que precisamde uma análise mais significativa, tais técnicas envolvem aspectos qualitativos e subjetivos não verificados neste trabalho.

Existe ainda vasta margem para exploração da Máquina Virtual do Android ART, que por sua vez é mais recente do que a Dalvik, pois ainda não foram realizados testes mais específicos para esta ferramentapela comunidade de desenvolvedores do Android.

7 REFERENCIAS

[1] VALLÉE-RAI, Raja. **Soot-a Java bytecode optimization framework**. 2000. 107f. Tese (MestradoemCiências) - University, Montreal. 2000.

[2] NIMODIA C.; Deshm R.**Android Operating System**. 2012. 13f. Dissertação - BabasehebNaik College of Engineering, India. 2012.

[4] ROMÁN, Manuel. A middleware infrastructure for active spaces. **IEEE pervasivecomputing**, v. 1, n. 4, p. 74-83, 2002.

[5]HANDSET ALLIANCE, O.: **Android.** Disponível em: <http://www.openhandsetalliance.com/android_overview.html>. Acessoem: 20 fev. 2015.

[6] ANDROID, S.: **Art e Dalvik**. Disponível em: <https://source.android.com/devices/tech/dalvik/> . Acesso em 22 fev. 2015.

[7] DEVELOPER, G. **Introducing ART**. 2014.disponível em <http://developer.android.com/training/articles/perf-tips.html> Acesso em 12 dez 2014.

[8] BRAHLER, Stefan.**Analysis of the android architecture**. 2010. 46f. Monografia - Universitat des Landes Baden-Wurttemberg. 2010.

[9] HISTÓRIA, JAVA. História **da Linguagem Java**. 2006.Disponível em:<http://www.htmlstaff.org/java/java15.php>. Acessado em, v. 11, 2006

[10] KURNIAWAN, Budi. **Java for Android**. United States of America, 2014.

[11]DEVELOPER, G. **PerformanceTips.** 2014.

disponível em <http://developer.android.com/training/articles/perf-tips.html> Acesso em 12 dez 2014.

[12] SERSON, Roberto RUBINSTEIN. **Programação orientada a objetos com Java 6.** Curso universitário. Brasport, 2008.

[13]BLOCH, Joshua. **Effective java (the java series).**United States of America,2008.

[14] CARVALHO, V. H. A. **Análise de impacto do uso de técnicas de programação no desempenho de aplicações android,** Trabalho de conclusão de curso (Engenharia de Software) – Universidade de Brasília – UnB, Brasília, 2014.

[15] TONINI, Aline. **Avaliação das boas práticas Android para desempenho.** Trabalho de conclusão de cuso (Ciência da Computação) – Universidade Federal de Pelotas, Rio Grande do Sul, 2013.

[16] PETRONI, Benedito Cristiano et al. AVALIAÇÃO DA USABILIDADE DA IDE ANDROID STUDIO. **Revista Eletrônica de Tecnologia e Cultura**, v. 1, n. 1, 2014.

[17] RABELLO, R. R.**Android: um novo paradigma de desenvolvimento móvel.**Revista WebMobile. 18 Ed. 2009.

[18] PETRY, Patrícia G. **Um sistema para o ensino e aprendizagem de algoritmos utilizando um companheiro de aprendizagem colaborativo**. 2005. 82f. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Santa Catarina. 2005.

[19] PARADA, Abilio Gambim; TONINI, Aline Rodrigues; DE BRISOLARA, Lisane Brisolara. **Geração Automática de Código Android Eficiente a partir de Modelos UML**. Universidade Federal de Pelotas, Brasil.

[20] SALIBA, W. L. C. **Técnicas de Programação – Uma Abordagem Estruturada**. São Paulo: Makron, McGraw-Hill, 1992.

[21]ALMEIDA, CAETANO N.**SISTEMA PARA AFINAÇÃO DE INSTRUMENTOS MUSICAIS ATRAVÉS DE TELEFONES CELULARES**. 2007. 85f. Monografia (Bacharelado em Sistemas de Informação) – Universidade do Planalto Catarinense. 2007.

[20] Leite, Mario. **Técnicas de Programação - Uma Abordagem Estruturada**. Rio de Janeiro: Brasport, 2006.