UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Graduação em Engenharia da Computação

Análise de Desempenho de Plataformas para Desenvolvimento com o Sistema Operacional Android

Júlio Gil da Fonte Freire

TRABALHO DE GRADUAÇÃO

Recife,

25 de abril de 2013

Universidade Federal de Pernambuco

Centro de Informática

Júlio Gil da Fonte Freire

Análise de Desempenho de Plataformas para Desenvolvimento com o Sistema Operacional Android

Trabalho apresentado ao Programa de Graduação em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Orientador: Ricardo Massa Ferreira Lima

Recife

25 de abril de 2013

Dedico este trabalho primeiramente a Deus, que sempre esteve ao meu lado. Dedico aos meus pais Júlio e Mônica, pelo apoio e amor que sempre me deram. Dedico aos meus irmãos Marcela, Raymundo e Rafael, à minha família e aos meus amigos e primos que sempre me apoiaram e acreditaram em mim.

Agradecimentos

Antes de qualquer coisa, gostaria de agradecer a Deus, que sempre esteve ao meu lado nos bons e maus momentos. Agradeço aos meus pais que sempre me deram a melhor educação possível, da mesa à faculdade, nenhum esforço foi poupado. Agradeço do fundo do coração. Em seguida, agradeço aos meus irmãos Marcela, Raymundo e Rafael pelas brigas e alegrias do dia a dia. Não sei o que seria de mim sem eles. Aos meus primos e familiares que, a pesar de se somarem as centenas, modelaram, cada um de seu jeito, o que eu sou hoje.

Agradeço aos meus colegas da faculdade, que as considerados hoje amigos de infância. Augusto, David, Marcelo, Breno, Julio, Hector, Luis, Tchelo. Em especial a Arthur, Pedro e Lais, que participaram ativamente do meu dia a dia na faculdade transformando o tolerável no incrível. Aos colegas de Ciências Palito, Vaca, Paulo e Victor que eu tanto brinquei e aperreei.

Agradeço aos colegas empreendedores do CITi, em especial Sotero, Flavio, Ed, e os canalhas Dudu, Farias e Rafa, com que eu tive orgulho de ter aprendido e ensinado, crescendo mais rapidamente do que jamais poderia ter imaginado.

Aos meus amigos irmãos Luciano, Ico, Bobinho, Assis, Zé Ivan, Elvis e vários outros com quem tive o prazer de compartilhar todos os momentos livres que tive direito, sempre atento para não falar sobre a faculdade.

Aos meus amigos da Elcoma, Claudio e Rafael, não só por sua ajuda direta com o projeto, mas pelo companheirismo e amizade do dia a dia.

Agradeço aos professores Sergio Cavalcanti e Hélio de Magalhaes por reacenderem a paixão pela engenharia. Ao professor Cristiano elas conversas empreendedoras. A professora Edna que foi uma verdadeira mãe, não só para mim, mas para toda a turma de Engenharia da Computação. A todos os professores do Centro de Informática pela excelente qualidade de ensino, que contribuiu com minha formação acadêmica.

E por fim, agradeço ao professor orientador e gerente dos projetos que fiz parte, Ricardo Massa pela colaboração, compreensão no meu trabalho de graduação e em toda sua contribuição no resto da minha vida profissional.

“Eu não posso mudar a direção do vento,
mas eu posso ajustar as minhas velas
para sempre alcançar o meu destino.”
Jimmy Dean

Resumo

A plataforma Android cresceu rapidamente desde o seu lançamento, não só em número de dispositivos, mas também na sua presença em mercados diferentes. Por causa da facilidade de se programar, flexibilidade da plataforma e confiança do mercado, empresas de base tecnológica tem adotado esta plataforma, embarcando sistemas com Android nas mais diversas áreas. No entanto, por ser majoritariamente presente em dispositivos móveis, para uso pessoal, a plataforma carece formas de analisar o desempenho de aplicações especificas. Este projeto propõe uma forma simples de as próprias empresas conseguirem desenvolver os testes necessários para escolher o dispositivo que melhor se adequará às suas necessidades, se baseando na aplicação lhe interessar. A empresa que adotar esta proposta saberá utilizar as ferramentas do Software Development Kit (SDK) do Android, ou Kit de Desenvolvimento de Software, e terá capacidade para simular testes e analisar os resultados possibilitando a tomada de decisão de qual o melhor dispositivo para a sua necessidade.

ABSTRACT

The Android platform has grown rapidly since its launch, not only in number of devices, but also in its presence in different markets. Because of the ease of programming, platform flexibility and confidence in the market, technology-based companies have adopted this platform, shipping with Android systems in several areas. However, being mostly present in mobile devices for personal use, the platform lacks ways to analyze the performance of specific applications. This project proposes a simple way the companies themselves can develop the necessary tests to choose the best device that will suit your needs, relying on the application of interest. The company that adopts this proposal will know how to use the tools of the Android Software Development Kit, and have the ability to simulate and analyze the test results enabling the decision making of which the best device for its needs.

Sumário

Índice de imagens

Índice de Tabelas

Índice de Equações

1 Introdução

1.1 Sistema Android

No cenário atual de plataformas moveis, o Android ("What is Android?", 2013) está entre as mais populares. De código aberto e completamente customizável, o sistema Android é uma estrutura de software abrangente que se divide em camadas: o Sistema Operacional, o Middleware e a camada de aplicação (Shabtai, Fledel, Kanonov, & Elo, 2009). Originalmente desenvolvido para dispositivos móveis, o sistema Android conseguiu rapidamente espaço no mercado de sistemas embarcados.

Desenvolvedores que queiram lançar aplicações para Android, podem submetê-las ao Google Play (Google Play, 2013), antigo Android Market, de onde usuários podem fazer o download delas e instalá-las. (Nauman & Khan, 2010).

Mantido pela Google, o sistema operacional é aberto, baseado em Linux, dividido em camadas e é compatível com diversos dispositivos. A flexibilidade e abertura do sistema fez com que milhares de pessoas contribuíssem para o desenvolvimento da tecnologia Android, enxugando e otimizando cada vez mais seus códigos.

1.2 A Escolha do Dispositivo Android

Com o crescente número de opções de dispositivos Android no mercado, cada vez mais as pessoas se perguntam “Qual o melhor dispositivo pra mim?”. No cenário de dispositivos celulares móveis, existem muitas fontes para se obter informações que respondam a essa pergunta. Vendedores em lojas físicas, comentários em lojas virtuais e recomendações de amigos, colegas e afins, são as primeiras opções de leigos.

Já para aqueles que têm conhecimento aprofundado das tecnologias que compõem um celular, essas recomendações não são sempre suficiente. Para uma análise mais aprofundada, é preferível recorrer a análises de desempenho ou comparativos entre dispositivos. Vários softwares de benchmark estão disponíveis para o sistema Android, como por exemplo o Quadrant Standard (Quadrant Standard no Google Play, 2013), o AnTuTu (AnTuTu Benchmark no Google Play, 2013) e o Vellamo (Vellamo no Google Play, 2013), que, apesar de possuírem limitações, cumprem bem seu papel de comparar as

características como Capacidade do CPU e barramento de memória dos dispositivos analisados.

1.3 Android para Indústrias de Base Tecnológica

O sistema Android foi anunciado publicamente no começo de 2008. Sendo considerada uma tecnologia consideravelmente nova, pelo fato de que ainda está sendo continua e rapidamente melhorada e atualizada tanto em suporte a novos componentes como em firmware, o Android tem ganho força tanto na indústria de telefonia móvel, quanto em outras industrias, como a de entretenimento automobilístico (Macario, Torchiano, & Violante, 2009) e mini computadores (Site Miniand, 2013). O crescimento acelerado da indústria vem de dois fatores: a natureza *Open-Souce* e a arquitetura do Android (Maia, Nogueira, & Pinho, 2013).

Sendo um projeto *Open-Source,* é possível analisar e entender o Android completamente, permitindo que seus componentes seja compreendidos, bugs sejam reparados, novas funções sejam propostas e adaptações para a adoção de um novo hardware sejam desenvolvidas. E por ser fundamentada no Kernel do Linux, o conhecimento já adquirido pela indústria pode ser reaproveitado.

No entanto, as indústrias de base tecnológica precisam de testes mais específicos para atender suas necessidades. Seja para levar em consideração testes em situações inóspitas, variar o teste de acordo com o encapsulamento do hardware, variar os testes com recursos diferentes dentro do mesmo conjunto de hardware, ou testar componentes específicos do conjunto de hardware, os softwares de benchmark disponíveis não fornecem nenhuma ferramenta para a tomada de decisão que envolva o desenvolvimento de um sistema embarcado.

Outro fator relevante na escolha do hardware para embarcar o sistema Android é o custo do dispositivo. Como o sistema funcionará com aplicações específicas, é possível e importante escolher previamente os recursos que comporão o dispositivo, a fim de não se gastar com componentes desnecessários.

1.4 Justificativa

Para dar novas opções de testes para as indústrias de base tecnológica, este projeto tentará mostrar uma forma simples para testar os dispositivos utilizando aplicativos desenvolvidos pela própria indústria.

Para dar suporte à análise dos resultados obtidos com os testes, este projeto também irá explicar o modelo de um Projeto de Experimento simples, no caso o Projeto de Experimento Fatorial 2^k (Montgomery, 2001), de tal forma que a indústria possa aplicar o projeto de experimentos sempre que necessário.

1.5 Objetivos

O objetivo geral deste trabalho é realizar, com auxílio de simulações, uma avaliação de desempenho e analise do sistema Android, demonstrando os passos de forma replicável para que indústrias de base tecnologia possam fazer os experimentos propostos in loco, e tenham condições de analisar seus resultados. O estudo será realizado no Emulador de Android do SDK fornecido pela Google, onde serão alteradas as configurações do sistema emulado, para fins de comparação. Para alcançar o objetivo geral, os seguintes objetivos específicos foram definidos:

- Estudar o modelo Projetos de Experimentos Fatoriais;
- Aprender o funcionamento do SDK (*Software Development Kit)* de Android, principalmente o ADT (*Android Developer Tool)*, o DDMS *(Dalvik Debug Monitor Server)*, o ADB *(Android Debug Bridge)* e o AVDM *(Android Vitrual Machine Manager);*
- Implementar uma aplicação simples para testar o desempenho dos dispositivos emulados;
- Analisar os resultados e estudar os efeitos das variações utilizando o Projeto de Experimentos Fatoriais;

1.6 Estrutura do trabalho

Este trabalho foi dividido da seguinte forma. No Capítulo 2, teremos uma visão geral de todos os pontos importante abrangidos neste trabalho. Começando pelo sistema Android, foi dado foco no seu funcionamento e arquitetura. A seguir é dada uma explicação sobre componentes que o compõe a análise de desempenho que será

utilizada neste trabalho, que são o Design de Projetos de Experimentos e o algoritmo aplicado para se realizar os experimentos. Ainda no Capítulo 2, será explicado o que vem a ser o teste de Lucas-Lehmer e como ele será utilizado neste trabalho. Concluindo o Capítulo 2, serão explicados os componentes do SDK de Android.

No Capítulo 3 a metodologia de análise de desempenho que foi explicada no anteriormente, será utilizada e aplicada na plataforma escolhida. Ainda neste Capítulo, será explicado como foram feitas as simulações e qual foi a metodologia para a obtenção dos dados dos testes. No Capítulo 4 serão apresentados os resultados dos testes, e posteriormente será feita a análise dos resultados obtidos.

2 Visão Geral

2.1 Android

2.1.1 Arquitetura

O Android (Site Android, 2013) é uma plataforma de software desenvolvida para dispositivos móveis em geral, como telefones celulares, tablets e sistemas embarcados. A plataforma é dividida em três camadas de software: o sistema operacional, a camada intermediaria, ou camada middleware, e uma camada de aplicações, como mostrado na Figura 1 Plataforma Android.

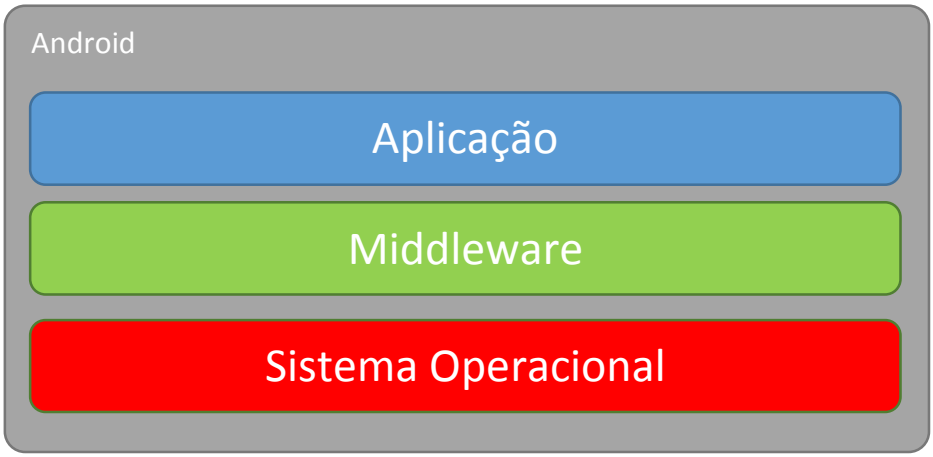

Figura 1 Plataforma Android

A arquitetura do sistema, mostrada na Figura 2 Arquitetura Android subdivide essas três partes em um total de cinco camadas, formalizando e modularizando sua estrutura. A camada de aplicação é a camada mais alta e ela se divide em camada de aplicação e camada de suporte de aplicações (ou *application framework)*. Nesta camada estão todos os programas nativos tais como Discador, Cliente de Email, Agenda, Câmera, Galeria de Imagens entre outros. O *framework* provê serviços genéricos para todas as aplicações, tais como gerenciador de janelas, e gerenciador de atividades. Ele também provê os serviços básicos de interface com as camadas inferiores, permitindo assim que desenvolvedores criem aplicações para a plataforma.

O *middleware* é composto por bibliotecas básicas em C/C++ e um *Android Runtime*. As bibliotecas são divididas em quatro categorias: *bionic libc*, biblioteca de funções, biblioteca de servidores nativos e biblioteca de abstração de hardware. Estas bibliotecas

proveem serviços como por exemplo renderização de imagens, gerenciamento de banco de dados SQLite, multimídia, funções padrões para navegação web, entre outras funções. Nestas bibliotecas estão contidas também funções responsáveis por garantir que o sistema funcione com o dispositivo que ele está instalado, abstraindo assim a preocupação da variação dos dispositivos nas camadas de aplicação.

Já no Runtime, encontram-se bibliotecas nativas de Java, chamadas de bibliotecas de núcleo, ou ainda *Core Libraries,* e a Máquina Virtual Dalvik, ou DVM. A maior parte das aplicações Android são escritas em Java, e são compiladas para o formato "*.class*". O sistema Android converte este arquivo posteriormente para o tipo *".dex"*, chamado de *Dalvik Executable,* que é executado na DVM. Cada aplicação Android tem é executada no seu próprio processo na própria instância da DVM.

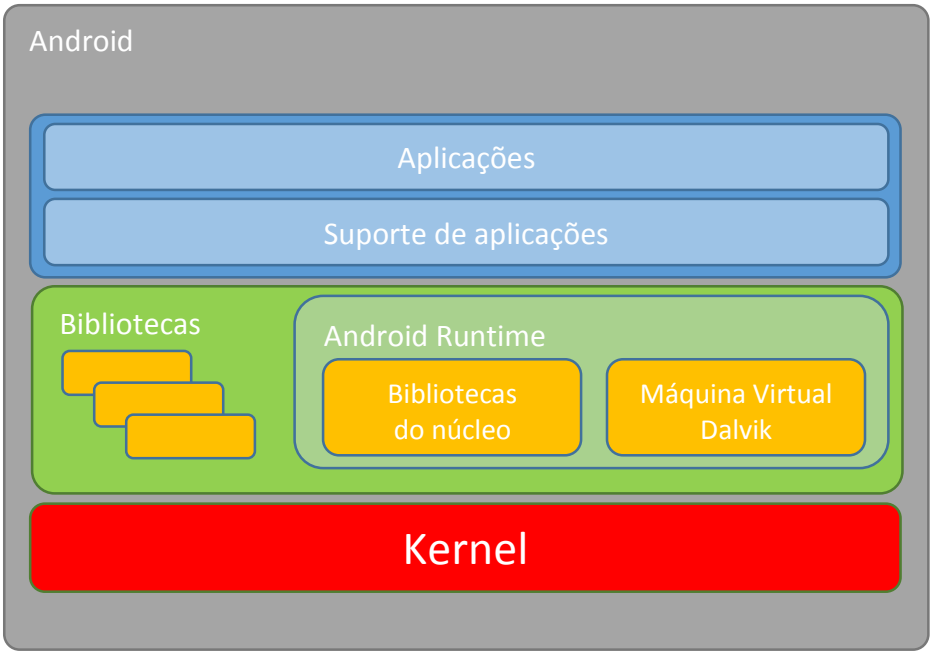

Figura 2 Arquitetura Android

A DVM é otimizada para baixo consumo de memória e projetada para suportar múltiplos processos de máquinas virtuais por dispositivo, com um interpretador de bytecode aprimorado, utilizando eficientemente a memória disponível no *Runtime*. A DVM também dá suporte a atividades do sistema operacional, tais como isolamento de processos, gerenciamento de memória e gerenciamento de threads.

A camada mais inferior é a do sistema operacional. Nesta camada estão os programas de gerenciamento de memória, configurações de segurança, gerenciamento de consumo energético, acesso ao sistema de arquivos, comunicação em rede de uma

forma geral, comunicação entre processos e os drivers, contidos no Kernel. O sistema Android é baseado no Kernel do Linux que é responsável por toda a comunicação com os Hardwares do dispositivo, como por exemplo o Bluetooth, Rede sem fio, a própria tela, botões, I/O e caixas de som.

2.1.2 Versões

Lançado em Setembro de 2008, o sistema Android já passou por diversas atualizações. Em cada uma delas, podem existir apenas pequenas correções de bugs, a adição de novas funcionalidades e troca completa de gerenciamentos internos de hardware e software. Quando há uma alteração mais significativa, o nível do API cresce, como controle para o desenvolvimento de novas aplicações, e na maior parte dos casos, um novo codinome é dado à versão.

Sem motivo aparente, os desenvolvedores do sistema Android apelidaram as verões do seu sistema com nomes de sobremesas cujas iniciais crescem em ordem alfabética. Começando na versão 1.5, codinome Cupcake, o sistema já passou pelas sobremesas: Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich e finalmente a mais atual, Jelly Bean, mostrados na Figura 3 Logos das versões de Android. A Tabela 4 do anexo A, mostra a relação dos codinomes com as versões e suas e o nível das APIs.

Figura 3 Logos das versões de Android

Na tabela também está a distribuição das versões de Android no mundo, de acordo com o (Dashboard Android, 2013). Posteriormente, neste trabalho, iremos

utilizar esta informação para simular dispositivos Android. Serão escolhidas as duas versões mais adotadas do Android, que são a Gingerbread, nível de API 10, mais especificamente a versão 2.3.3 e a Ice Cream Sandwich, nível de API 15, versão 4.0.3.

2.2 Design e Análise de Experimentos

2.2.1 Projeto de Experimento

Um projeto experimental formal procura obter as informações com um número mínimo de experimentos. O objetivo é reduzir o tempo gasto no trabalho de junção dos dados. Geralmente este projeto precisa de uma análise formal dos experimento para detectar e separar os efeitos que podem afetar o desempenho do sistema, determinar se um fator tem um efeito significante ou se a diferença observada é simplesmente devido a variações randômicas causadas por parâmetros e erros de medidas que não foram controlados.

De acordo com (Jain, 1991) um projeto experimental formal geralmente possui os seguintes componentes:

- **Variável de Resposta**: nome dado ao resultado do experimento. Geralmente é a medida do desempenho do sistema. Tarefas completadas por intervalo de tempo, tempo de execução são dois dos exemplos mais comuns de Variáveis de Resposta.
- **Fatores**: nome dado a uma variável que afeta a variável de resposta. Exemplos comuns na comutação são tamanho da memória RAM ou clock da CPU.
- **Níveis**: nome dado aos valores que um fator pode assumir. Exemplos de níveis de memória RAM são 512MB, 1GB e 2GB. Cada nível constitui uma alternativa para o fator.
- **Fatores Primários**: Os fatores cujos efeitos precisam ser quantificados são chamados de primários, ou principais. Exemplificando num computador, os fatores mais importantes a serem estudados seriam o número de núcleos numa CPU, o tamanho da memória e o número de discos. Logo, este projeto tem 3 fatores principais.

- **Fatores Secundários**: São fatores que podem afetar o resultado na performance do sistema, mas não se tem interesse em estuda-los.
- **Replicações**: Nome dado ao número de vezes que o experimento será feito. Se um algoritmo precisar ser rodado 4 vezes para testar o desempenho de um sistema, dizemos que o experimento teve 4 replicações.
- **Projeto**: Consiste na especificação do número de experimentos, das combinações dos níveis dos fatores para cada experimento e do número de replicações. Exemplificando com os exemplos citados acima, temos um experimento com três fatores principais, que podem assumir níveis diferentes. CPU pode ter 1, 2 ou 4 núcleos; Memória RAM pode ter 512MB, 1GB, 2GB ou 4GB; e é possível ter 1 ou 2 discos. Queremos ainda executar 4 vezes o algoritmo. Finalmente veremos que serão necessárias 1x3x4x2x4 = 96 observações para realizar este projeto.
- **Unidade Experimental**: Qualquer entidade que é usada para o experimento é chamada de uma unidade experimental. Por exemplo, o computador que está executando o sistema, enquanto as medições estão sendo executadas, pode ser considerado uma unidade experimental.
- **Interação**: diz-se que dois fatores A e B interagem se os efeitos de um dependem do nível de outro.

2.2.2 Projeto de Experimento Fatorial 2^k

Projetos fatoriais são usados extensivamente em experimentos envolvendo fatores onde é necessário estudar o efeito conjunto dos fatores na Variável de Resposta. No projeto 2^k fatorial, cada um dos k fatores assumem apenas 2 níveis. Esses níveis podem ser quantitativos, como por exemplo, dois valores de tamanho de memória, ou de temperatura ambiente. Ou podem ser qualitativos, como por exemplo, algum outro algoritmo sendo executado ou não ao mesmo tempo durante o experimento, ou a presença ou não de algum componente. A esse tipo de projeto fatorial, é dado o nome de projeto de experimento fatorial 2^k. (Montgomery, 2001)

O projeto fatorial 2^k é extremamente útil nas fases inicias de um projeto, quando ainda se tem muitos fatores para serem investigados. 2 é o menor número de variações

que um fator pode ter, logo é possível ver logo no começo do projeto, quais fatores serão relevantes no futuro do estudo e quais não afetam significativamente.

Os passos seguidos para a aplicação desta metodologia são:

- Estimar quais fatores podem causar efeito no sistema;
- Conceber um modelo de desempenho ideal;
- Realizar os testes estatísticos;
- Refinar o modelo, retirando variáveis não significantes, se necessário;
- Fazer análise residual para verificar a adequação do modelo e suas suposições e;
- Interpretar os resultados obtidos.

Se o projetista julgar que os resultados deste projeto foram insatisfatórios, ele é incentivado a expandir suas pesquisas e seus experimentos. No entanto, esta projeto, mesmo que insuficiente, terá sido de grande auxílio para o pesquisador, ao mostrar exatamente em quais fatores ele deve focar e se aprofundar mais, estudando situações com mais de dois níveis.

2.3 Lucas Lehmer e os Primos de Mersenne

2.3.1 Definições

Para se realizar os testes de estresse foi escolhido o teste matemático de primalidade de Lucas-Lehmer. Criado em 1856 e otimizado em 1878 e 1930, o teste de Lucas-Lehmer tem como objetivo dizer se, dado número primo *P*, o número 2^P-1 é um número primo, ou seja, é um primo de Mersenne. Um primo de Mersenne é um primo da forma:

$$M_P = 2^P - 1 \qquad \textit{Eq 2.1}$$

Tal que *P* também seja um número primo.

Apesar de ter uma forma simples, estes primos são muito raros, tendo sido encontrados até hoje apenas 48 deles, sendo o número $2^{57.885.161}$-1 o maior primo já encontrado e também o maior primo de Mersenne já encontrado. O teste de Lucas-Lehmer propõe um algoritmo simplificado para testar se o dado número primo é de fato

um primo de Mersenne, sem ter que testar todos os números que são menores que ele, abordagem conhecida como Força Bruta, ou Busca Exaustiva.

O teste de Lucas-Lehmer funciona da seguinte forma. Definida a sequência $\{s_i\}$ para todo i≥0 da seguinte forma:

$$s_i = \begin{cases} 4 & se\ i = 0 \\ s_{i-1}^2 - 1 & se\ i > 0 \end{cases} \qquad Eq\ 2.2$$

O número M_P, onde *P* é um número primo, é um primo de Mersenne se, e somente se

$$s_{P-2}\ (mod\ M_P) = 0 \qquad Eq\ 2.3$$

A prova deste teorema pode ser encontrada em (J.W.Bruce, 1993).

O algoritmo proposto para executar o teste de Lucas-Lehmer segue o seguinte pseudocódigo:

```
Lucas-Lehmer (p)
      var s = 4
      M = 2^p -1
      repetir p-2 vezes
            s <- ((s x s) - 2) mod M
      se s for igual a 0
            é primo
      senão
            é composto
```

Figura 4 Pseudocódigo do Algoritmo de Lucas-Lehmer

Onde *p* é um número primo dado e *M* é potencialmente o primo de Mersenne M_P. A variável *s* representa os valores da sequência $\{s_i\}$, no entanto, no loop que será executado *p-2* vezes, s_i não recebe o valor de $s^2_{i-1} - 2$, mas sim $s^2_{i-1} - 2\ (\text{mod } M_P)$. Esta variação garante ainda que o valor de *s* nunca será equivalente ao de um inteiro com mais do que *p* bits, simplificando assim os cálculos.

2.3.2 O Prime 95

O maior primo de Mersenne número, que é $2^{57.885.161}-1$ e seus 12 predecessores foram descobertos utilizando a ferramenta Prime95 (Site Prime95, 2013). Esta ferramenta foi desenvolvida com o intuito de descobrir números primos de Mersenne. No entanto, mesmo com todas as otimizações do algoritmo de Lucas-Lehmer, descobrir números primos exige capacidade de processamento e armazenamento grandes.

Foi percebido que ao se executar o Prime95, o computador disponibilizava todos os seus recursos de CPU e memória RAM para executar o software. Muitos entusiastas começaram a utilizar esta ferramenta para testar a integridade dos seus equipamentos, uma vez que era uma forma simples e direta de forçar o teste dos componentes do seu computador simultaneamente. Como as iterações do algoritmo de Lucas-Lehmer são interdependentes, nada pode falhar durante a execução dos testes, por isso utilizar um software que garantia a corretude da sua execução é importante. Os criadores do Prime95 acabaram por desenvolver uma parte extra do software, chamada de *"Torture Test Mode"*, que permite que o usuário execute o algoritmo por um tempo determinado e escolhendo que quantidade de recurso ele quer testar.

2.3.3 Adaptação para análise de desempenho

Considerando que o teste de Lucas Lehmer consome bem os recursos do dispositivo que ele está rodando, e pela sua facilidade de compreensão e implementação, este teste foi adaptado para ser utilizado em analises de desempenho.

Ao mudar a abordagem de num dado tempo, ver quantos números primos ele consegue encontrar, o algoritmo implementado para estre trabalho deixa um intervalo fixo definido onde serão testados todos os valores para se descobrir quantos e quais deles são primos de Mersenne. O tempo que o algoritmo tomar para fazer este teste será a variável de resposta do nosso projeto de experimento.

A implementação do algoritmo se encontra no Anexo B e será explicada em mais detalhes posteriormente na seção 3.2.3.

2.4 SDK

O Software Development Kit (SDK) do Android fornece aos desenvolvedores todas as ferramentas para ele desenvolver um aplicativo para Android. Desde o ambiente de desenvolvimento, até ferramentas para gerenciar o download e instalação de novos pacotes e configurações para as versões mais atuais do Android.

Para este projeto, foram utilizadas as seguintes ferramentas:

2.4.1 ADT

O *Android Developer Tools* (ADT), ou Ferramentas para o Desenvolvedor Android, é um plugin para o Eclipse (Site Eclipse, 2013) que integra nesta IDE todas as ferramentas do SDK. Ele oferece vários recursos que ajudam a desenvolver aplicações para Android rapidamente e com segurança. Além da integração com o Eclipse, que é

uma IDE bem estabelecida e conhecida pelos desenvolvedores, o ADT oferece ferramentas visuais para diversos comandos do SDK, que geralmente seriam acessados só através de linhas de comando (Figura 5).

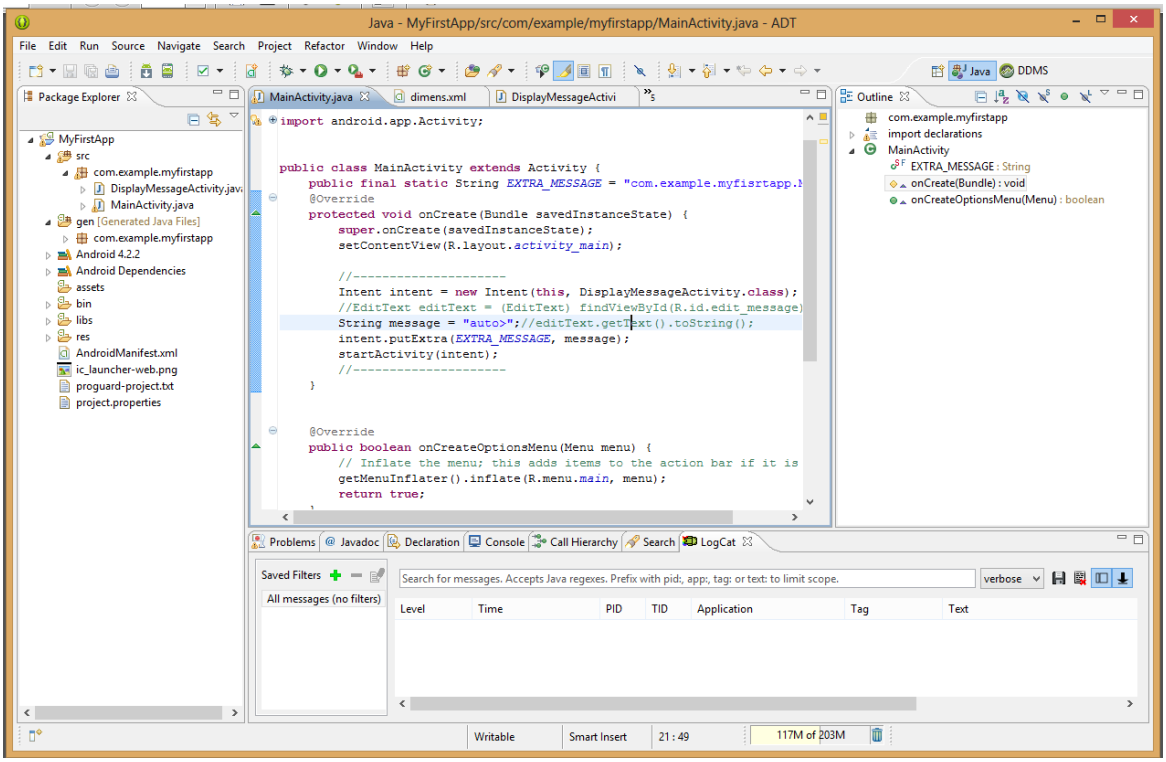

Figura 5 Android Developer Tool

O ADT também oferece ferramentas para desenhar e prototipar a interface da aplicação. Com editores de texto e interpretadores XML, o ADT exibe a aparência da aplicação enquanto o desenvolvedor cria a aplicação (Figura 6). Se o desenvolvedor

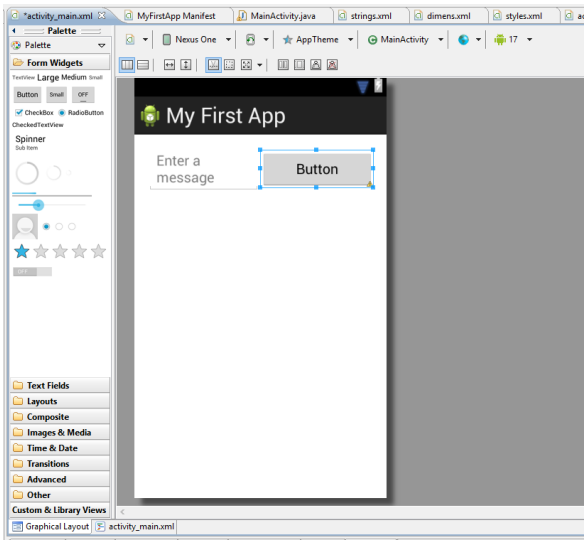

Figura 6 Ferramenta gráfica para UI

preferir, ainda pode criar a interface graficamente e depois integrar o código automaticamente gerado com as funções que implementar.

O ADT facilita a vida do desenvolvedor por ter interface simples, ter capacidades de gerenciar o projeto que está sendo desenvolvido, e ser a porta de acesso para todos os componentes do SDK.

2.4.2 DDMS

Os dispositivos Android vem nativamente com uma ferramenta de depuração chamada Dalvik Debug Monitor Server (DDMS), ou Servidor de Monitoramento de Debug do Dalvik, que fornecem serviços de captura de tela, informações de Heap e Threads do dispositivo, estado do LogCat, processos e intensidade do sinal recebido pela antena do dispositivo, e ainda coleta informações de chamadas, mensagens de texto e dados de localização.

No Android, cada aplicativo é executado em seu próprio processo, cada um dos quais é executado em sua própria máquina virtual (VM). Cada VM expõe uma única porta que um depurador pode anexar. Quando DDMS é iniciado, ele se conecta ao ADB. Quando um dispositivo for conectado ao computador, um serviço de monitoramento de VM é criada entre ADB e o DDMS, que notifica o DDMS quando uma VM no dispositivo é iniciado ou encerrado. Uma vez que a VM está rodando, DDMS recupera ID do processo da VM (PID), via ABD, e abre uma conexão com depurador do VM, através do daemon do ABD (ADBD) do dispositivo. DDMS pode agora falar com a VM usando um protocolo personalizado.

2.4.3 ADB

Android Debug Bridge (ADB) é uma ferramenta de linha de comando versátil que permite a comunicação com uma instância do emulador ou dispositivo Android conectado ao computador via USB. Trata-se de um programa de cliente-servidor, que inclui três componentes:

- Um cliente, que funciona em seu computador. Você pode invocar um cliente a partir de um shell mediante a emissão de um comando de ADB. O ADT já cria automaticamente um cliente ADB e mostra diversas informações na sua interface.

- Um servidor, que é executado como um processo em segundo plano em sua máquina de desenvolvimento. O servidor gerencia a comunicação entre o cliente e o daemon do ADB rodando em um emulador ou dispositivo.
- O daemon, que é executado como um processo de fundo em cada emulador ou instância do dispositivo.

A informação mais importante adquirida através de um ADB para este projeto foi a do LogCat, que será explicado a seguir.

2.4.4 Emulador Android

O SDK do Android inclui um emulador de dispositivos celulares, ou seja, um dispositivo móvel virtual que é executado no seu computador. Este emulador é uma ferramenta que fornece um dispositivo móvel virtual em que você pode executar seus aplicativos Android. Ele simula o sistema Android em sua totalidade, até o nível de kernel, que inclui um conjunto de aplicativos pré-instalados (como o discador) que podem ser acessados a partir de suas aplicações. O emulador simula todos os recursos de Hardware e Software de um dispositivo móvel típico em tudo, exceto na capacidade de fazer chamadas. O dispositivo emulado conta com todos os botões padrões de Android e ainda com um teclado virtual, se o desenvolvedor quiser. Na tela simulada do dispositivo, o mouse funciona como interface de toque (Figura 7 AVD).

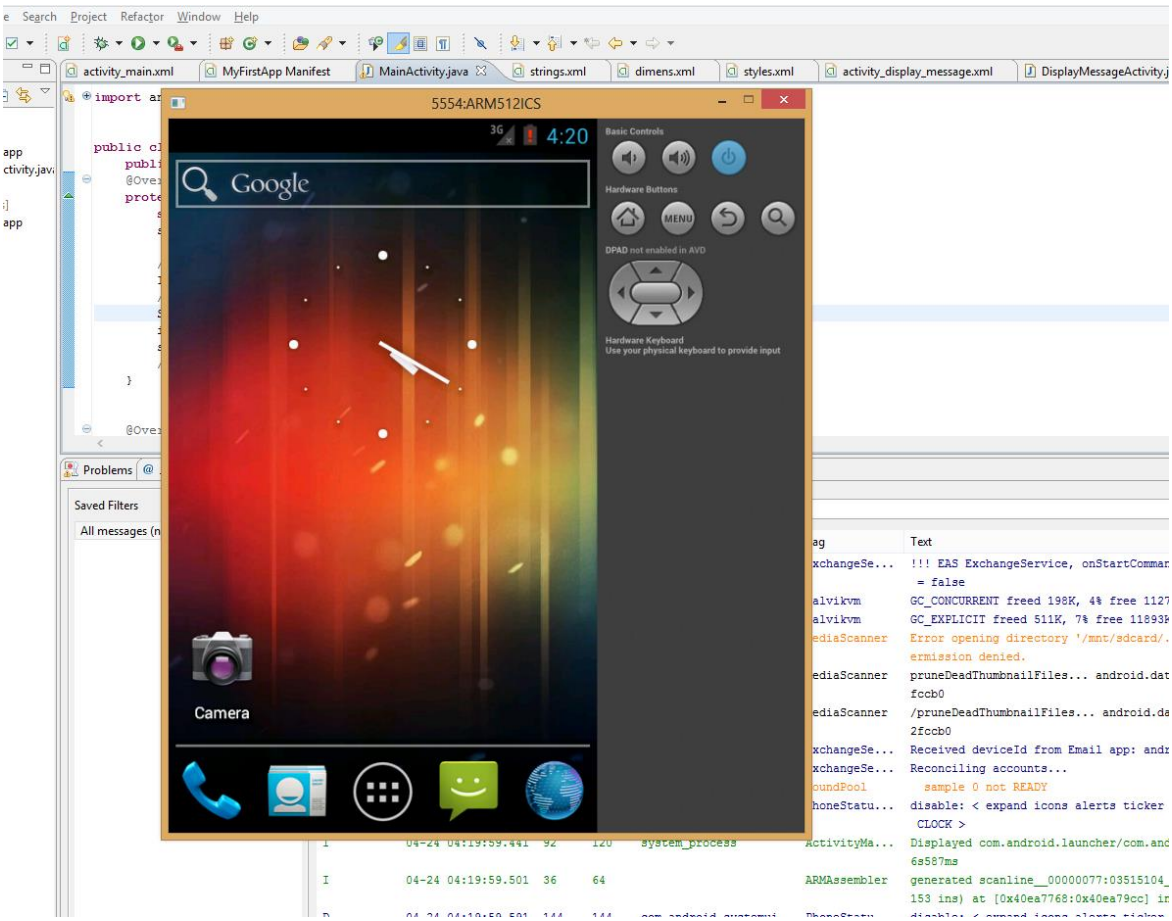

Figura 7 AVD

Para configurar um dispositivo virtual, o SDK de Android conta com uma ferramenta chamada Android Virtual Device (AVD) Manager. Nela é possível configurar quanto de memória seu dispositivo virtual terá, qual será o tamanho de sua tela, qual o processador, quanto de armazenamento externo, qual a versão do Android que será executada no dispositivo, entre outras.

2.4.5 LogCat

O sistema de Log Android fornece um mecanismo para coleta e visualização de saída de depuração do sistema. Registros de várias aplicações e partes do sistema são recolhidos numa série de buffers circulares, que podem então ser visualizados pelo comando *logcat.*

O comando *logcat* é invocado a partir de um shell ADB, mas como cientes ADB são automaticamente criados no ADT, enquanto se estiver executando alguma aplicação

ou simplesmente monitorando o sistema do dispositivo conectado ou emulado, é possível ver na interface da IDE o log em tempo real (Figura 8).

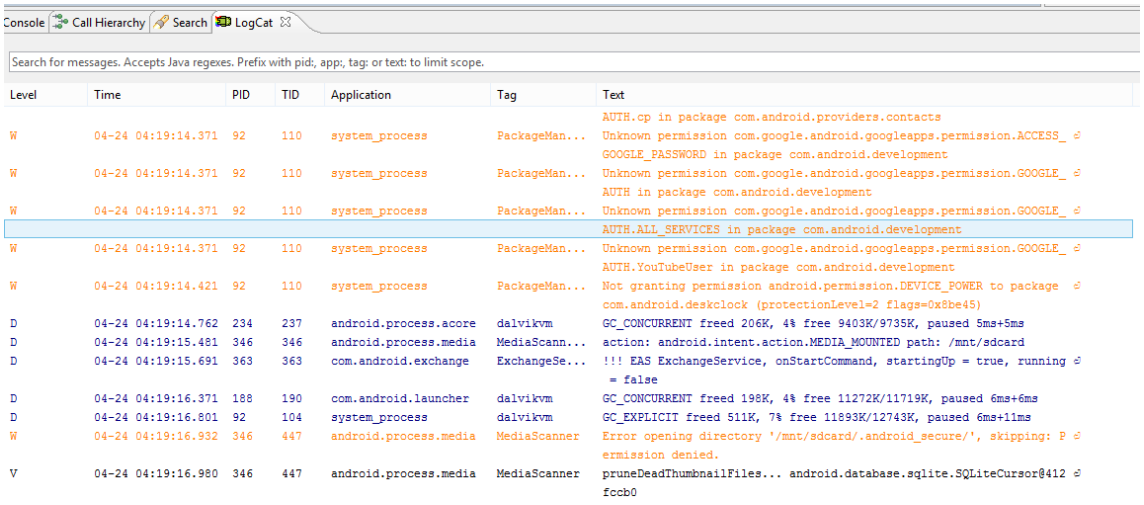

Figura 8 LogCat

3 Metodologia Empregada

3.1 Experimento fatorial 2^k

Para realizar o projeto de experimentos foram determinados os seguintes componentes:

- **Variável de Resposta**: Tempo de execução dos algoritmos de Lucas-Lehmer (Figura 4) para *p* variando de 2 a 1000, e de manipulação de arrays de bytes longos. Ambos os algoritmos serão explicados em detalhes na seção 3.2.3;
- **Fatores**: O sistema Android tem um número grande de fatores que podem influenciar na sua performance, como a arquitetura, frequência do clock e número de núcleos de um processador, barramento e quantidade de memória, versão do Android, tamanho dos discos interno e externo, resolução da tela, entre vários outros. Este trabalho, no entanto, irá trabalhar só com o emulador de AVDs, que tem um número menor de fatores. São eles: arquitetura do processador, tamanho da memória, tamanho do Heap, espaço de armazenamento interno, espaço de armazenamento externo, dispositivo base e versão do Android;
- **Fatores Primários**: O objetivo deste trabalho é ensinar a indústria como escolher o melhor dispositivo Android para a sua aplicação. Tendo isto em mente, escolhemos os fatores mais relevantes para a tomada de decisão inicial de um projeto que são a arquitetura do processador, tamanho da memória e versão do Android;
- **Fatores Secundários**: O dispositivo base foi mantido o mesmo, como mencionado na seção 3.2.2, o tamanho do Heap fixo em 32MB, pois é um tamanho aceito por ambos as versões de Android escolhidas para fazer o teste. O armazenamento interno e o externo foram inalterados, e ambos mantidos em 200MB.
- **Níveis**: As arquiteturas de CPU testadas foram a ARMv7 da série Cortex-A (ARMv7, 2013) e a Atom x86 da Intel (Intel Atom, 2013). Os tamanhos de memória escolhidos foram 256MB e 512MB. E por fim, as versões utilizadas foram a Gingerbread versão 2.3.3 e a Ice Cream Sandwich 4.0.3, por serem as mais adotadas mundo afora, de acordo com a tabela do Anexo A.

- **Replicações**: Foram realizadas 10 repetições de um teste preliminar no emulador de Android, com 256MB de memória RAM, processador ARMv7 e versão do Android 4.0.3; Em cada uma das repetições, era executado o algoritmo explicado na seção 3.2.3, com *p* variando de 2 a 500. Os resultados obtidos foram:

12046 ms	11024 ms	11454 ms	10951 ms	11252 ms
11127 ms	11141 ms	10980 ms	11137 ms	11005 ms

Tabela 1 - Resultados teste preliminar

A média $\overline{x}$ dos valores deste experimento foi

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \textit{Eq 3.1}$$

$$\overline{x} = 11211{,}7\ ms \qquad \textit{Eq 3.2}$$

Onde x_i é o valor da variável de resposta para cada repetição do experimento e n é o número de repetições. Calculamos o desvio padrão amostral seguindo a seguinte fórmula:

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2} \qquad \textit{Eq 3.4}$$

$$s = 11587{,}4\ ms \qquad \textit{Eq 3.3}$$

Considerando que queremos que nosso nível de confiança seja de 95%, temos que o escore Z na tabela de distribuição normal é 1,895. A margem de erro ε foi de 0,4%;

Calculamos então o número de replicações *N* de acordo com a seguinte fórmula:

$$N = \left(\frac{Z * s}{\overline{x} * \varepsilon}\right)^2 \qquad \textit{Eq 3.6}$$

$$N = 36{,}46 \cong 40 \qquad \textit{Eq 3.5}$$

Logo o projeto terá para cada configuração dos fatores 40 replicações.

- **Projeto**: Dado que o projeto tem 3 fatores primários, e 40 replicações, o projeto terá um total de 2x2x2x40 = 320 observações.

- **Unidade Experimental**: Todos os testes foram realizados no emulador de dispositivos virtuais, como mencionado na seção 3.2.2. Foram criadas 8 AVD com todas as combinações possíveis dos fatores escolhidos.
- **Interação**: A interação dos fatores será observada e comentada na seção encontrada. Estas interações deverão ser consideradas destaque na análise dos resultados, pois é baseada nelas que a decisão sobre qual a melhor plataforma é feita.

3.2 Simulações

Os experimentos realizados para este trabalho tem a finalidade de serem didáticos. A intenção é mostrar para a indústria como replicar o experimento in loco, sabendo como analisar e escolher o seu dispositivo mais adequado.

Foram realizados testes com dois algoritmos simples e três fatores foram escolhidos para serem variados. A seguir é mostrado como criar uma AVD no gerenciador e como variar os componentes desejados.

3.2.1 Computador e condições

Como o simulador depende da máquina que ele está rodando, é importante que todos os testes estejam na mesma condição de uso do computador para que a comparação entre os resultados seja válida.

Todos os testes foram realizados no mesmo laptop, com processador Intel® Core™ i7-2670QM, com 8GB de memória RAM, Sistema Operacional Windows™ 8. Todos os experimentos foram realizados enquanto o computador estava ligado na tomada e com o modo de energia configurado para alta performance.

3.2.2 Emulador

Para realizar os testes, é necessário saber o que se pode e o que se quer variar. No gerenciador de AVDs temos acesso a essas informações (Figura 9). Na interface vemos que podemos alterar além do nome, a versão do Android, a CPU utilizada, quantidade de memória RAM, se vai ser necessário emular um teclado, quantidade de memória de armazenamento interna e externa, entre outras.

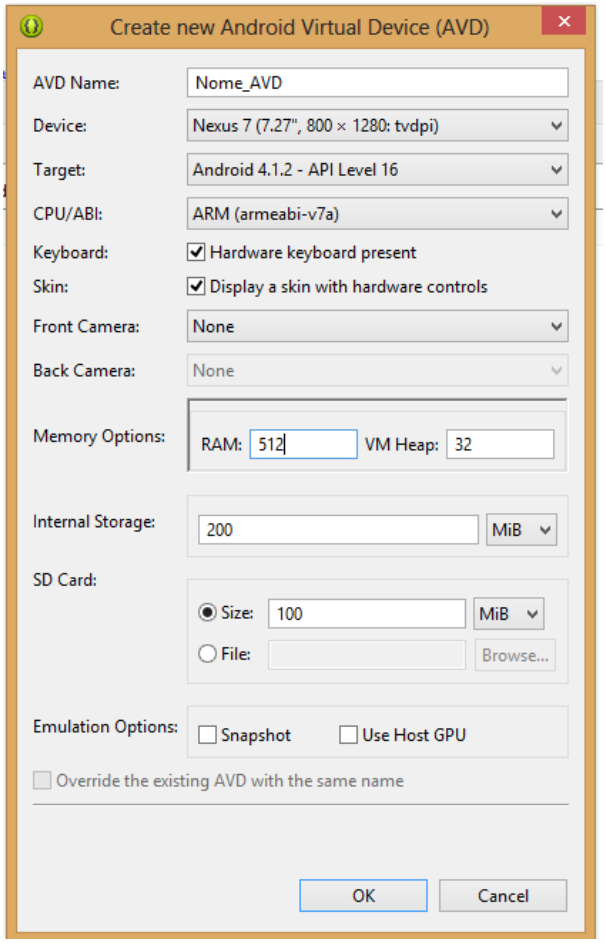

Figura 9 AVD Manager

Para este projeto, foram foi decidido que os fatores a serem variados serão a CPU que poderá ter a arquitetura ARM, ou uma arquitetura x86. A quantidade de memória RAM, que variará entre 256MB e 512MB, e finalmente a versão do Android, que baseado na Tabela 4 do Anexo B Código dos testes, será ou a versão Gingerbread 2.3.3, ou a versão Ice Cream Sandwich 4.0.3.

No decorrer do projeto, foi descoberto que o emulador não dá suporte à versão Gingerbread na arquitetura do processador Atom x86. Por isso, duas configurações de dispositivos não foram utilizadas.

No gerenciador de AVDs existe também uma opção que tem celulares padrões que podem ser usados como base para a criação do simulador. Para garantir igualdade nos testes, todas as AVDs criadas foram baseadas no Nexus S.

Por fim, para garantir a padronização dos testes, evitar retrabalho e organizar melhor o projeto de experimento, todas as 6 AVDs (2^3 combinações dos fatores, menos 2 combinações sem suporte) foram criadas antes de ser realizado qualquer teste. O nome de cada AVD segue o seguinte padrão: [CPU][MEM][VER], onde CPU pode ser x86 ou ARM; MEM pode ser 256 ou 512; e VER pode ser GB (Gingerbread) ou ICS (Ice Cream Sandwich).

3.2.3 Aplicação Android

Para se testar efetivamente o desempenho do sistema, foram desenvolvidos dois algoritmos, um para estressar o processador e o outro para estressar memória. Os algoritmos foram desenvolvidos de forma simples e direta, só para uso controlado neste experimento. Mais uma vez enfatizando o caráter didático deste trabalho, deve-se mencionar que os algoritmos foram criados também para exemplificar o uso do SDK de Android e mostrar que tipo de informação é possível comparar.

O primeiro deles é baseado no teste de Lucas Lehmer, e foca no estresse da CPU, ao executar diversas operações aritméticas relativamente custosas para o processador. Como mencionado anteriormente, o teste de Lucas Lehmer serve para encontrar números primos de Mersenne. Neste algoritmo, no entanto, foi definido um valor fixo, no caso 1000, e o algoritmo testava todos os números entre 2 e este valor fixo para testar se tal número atendia as especificações de Lucas Lehmer. Como o valor era fixo, o tempo de execução dessas 999 iterações varia de acordo com o sistema testado, por isso este foi o parâmetro escolhido para a primeira análise de desempenho.

O segundo algoritmo faz uso do máximo de memória possível para uma aplicação Android. Ao criar um array de 8*1024*1024 bytes (8MB), com números aleatórios, e operar sobre este array também de forma aleatória, o algoritmo garante que a memória estará sendo usada tanto quanto possível. Por ter esta limitação de uso da memória, é pouco provável que a variação do tamanho de memória afete o resultado deste teste,

no entanto a forma de se gerenciar a memória, que é diferente entre versões do Android, pode desempenhar um papel importante na performance deste teste.

3.3 Obtenção dos Dados

Antes de iniciar cada teste, a AVD que será testada é iniciada no gerenciador. Quando ela for devidamente iniciada, verifica-se se ela foi detectada pelo ADT através do DDMS.

Uma vez detectada, define-se no código da aplicação quantas iterações serão realizadas para aquele teste e qual teste será realizado. Após alteradas essas informações, executa-se a aplicação no dispositivo.

Quando o dispositivo terminar de executar o algoritmo, deve-se pegar as informações no logcat referentes ao tempo de execução do algoritmo. A classe TimingLogger() foi utilizada para fazer a contagem do tempo foi utilizado em cada algoritmo. O formato de saída no logcat dessa classe está representado na Figura 10. Para cada AVM testada, foi salvo um arquivo de texto com o mesmo nome da AVM, contendo o texto fornecido por essa classe.

```
D/TAG     ( 3459): methodA: begin
D/TAG     ( 3459): methodA:       9 ms, work 1
D/TAG     ( 3459): methodA:       1 ms, work 2
D/TAG     ( 3459): methodA:       6 ms, work 3
D/TAG     ( 3459): methodA: end, 16 ms
```

Figura 10 Saída TimingLogger()

4 Resultados

Após feitas todas as observações, o próximo passo é analisar as informações obtidas.

4.1 Resultados e contribuições

Os resultados do primeiro teste se encontram no Anexo C Resultados do Teste 1 (Lucas Lehmer) e estão resumidos na Tabela 2.

	CPU	Mem	Versão	Média Tempo de resposta (ms)		Desvio Padrão Tempo de resposta (ms)
1	ARM	256MB	2.3.3	52.775	±	3.487
2	ARM	256MB	4.0.3	46.892	±	2.308
3	ARM	512MB	2.3.3	51.170	±	2.038
4	ARM	512MB	4.0.3	47.679	±	937
5	x86	256MB	2.3.3	----	±	----
6	x86	256MB	4.0.3	34.996	±	483
7	x86	512MB	2.3.3	----	±	----
8	x86	512MB	4.0.3	36.318	±	1.603

Tabela 2 Resumo resultados teste 1

A partir destes resultados, é importante entender qual o efeito de cada variação dos fatores na variável de resposta que é a média. Podemos ver inicialmente que a melhor configuração foi a da plataforma x86 com 256MB de RAM e utilizando a versão Ice Cream Sandwich do Android, mas não sabemos quão relevante é essa melhora em relação às outras configurações e qual a influência da variação de cada fator.

De acordo com (Jain, 1991) é possível calcular a contribuição de cada fator baseado no seguinte conjunto de equações. Digamos que a variável de resposta y se relaciona da seguinte forma com os fatores:

$$y = x_0 + q_A * x_A + q_B * x_B + q_{AB} * x_A * x_B \quad \textit{Eq 4.1}$$

Onde q_F é a contribuição do Fator F e

$$x_F = \begin{cases} -1 & se\ Fator\ F\ assumir\ valor\ baixo \\ 1 & se\ Fator\ F\ assumir\ valor\ alto \end{cases} \quad \textit{Eq 4.2}$$

Substituindo os possíveis valores da Eq 4.2 na Eq 4.1, teremos as seguintes equações:

$$y_1 = q_0 - q_A - q_B + q_{AB} \quad \textit{Eq 4.6}$$

$$y_2 = q_0 + q_A - q_B - q_{AB} \quad \textit{Eq 4.5}$$

$$y_3 = q_0 - q_A + q_B - q_{AB} \quad \textit{Eq 4.4}$$

$$y_4 = q_0 + q_A + q_B + q_{AB} \quad \textit{Eq 4.3}$$

Onde os y_i são as respostas do sistema para cada uma das configurações. Resolvendo essas equações para os q_F's, temos:

$$q_0 = \frac{1}{4}(y_1 - y_2 - y_3 + y_4)$$

$$q_A = \frac{1}{4}(-y_1 + y_2 - y_3 + y_4) \quad \textit{Eq 4.7}$$

$$q_B = \frac{1}{4}(-y_1 - y_2 + y_3 + y_4)$$

$$q_{AB} = \frac{1}{4}(y_1 - y_2 - y_3 + y_4)$$

Como não temos os resultados das configurações com processador x86 e versão Gingerbread, faremos o estudo dos efeitos dos fatores duas vezes. Uma mantendo o processador ARM, e outra mantendo a versão Ice Cream Sandwich.

Mantendo o processador ARM, temos que

$$y = 49.628 - 204 * q_{MEM} - 2.343 * q_{VER} + 598 * q_{MEM} * q_{VER} \quad \textit{Eq 4.8}$$

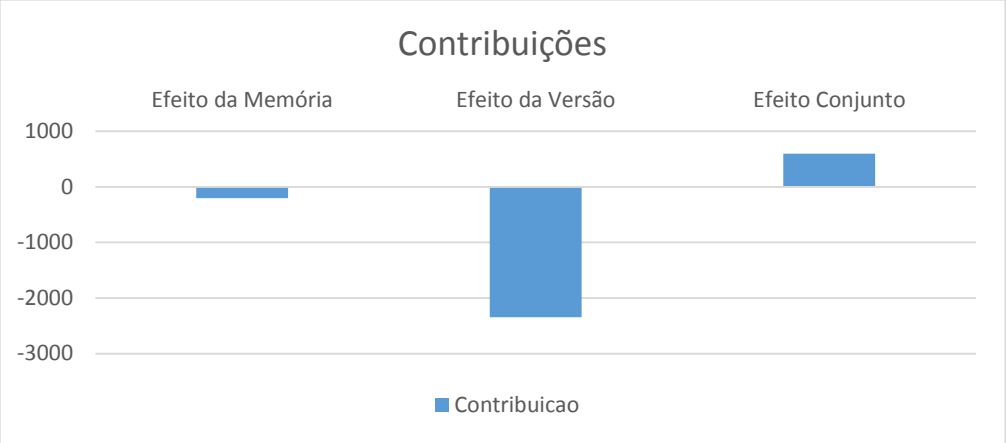

Figura 11 Contribuições absolutas Memoria e Versão Teste 1

Pela Figura 11 já podemos perceber que o efeito da versão escolhida, para o Teste 1, é muito maior do que o efeito do tamanho da memória. Para calcular com precisão esta influência, (Jain, 1991) a seguinte formula. Seja T a soma do quadrado das variações:

$$T = 2^2.q_A^2 + 2^2.q_B^2 + 2^2.q_{AB}^2 \qquad \textit{Eq 4.9}$$

A contribuição de cada fator será dada por

$$Contribuição\ de\ F = \frac{2^2.q_F^2}{T} \qquad \textit{Eq 4.10}$$

Aplicando os valores das contribuições expostos na Eq 4.8, temos que a contribuição da memória é de diminuição do tempo de execução e representa 1% da variação, a contribuição versão do Android é também de diminuição e representa 93% e a influência de um no outro contribui com um aumento do tempo de execução e representa com 6%.

Em seguida, mantendo a versão do Android na Ice Cream Sandwich, e variando o tamanho da memória e a arquitetura do processador, temos:

$$y = 41.471 - 5.814 * q_{CPU} + 527 * q_{MEM} + 133 * q_{CPU} * q_{MEM} \qquad \textit{Eq 4.11}$$

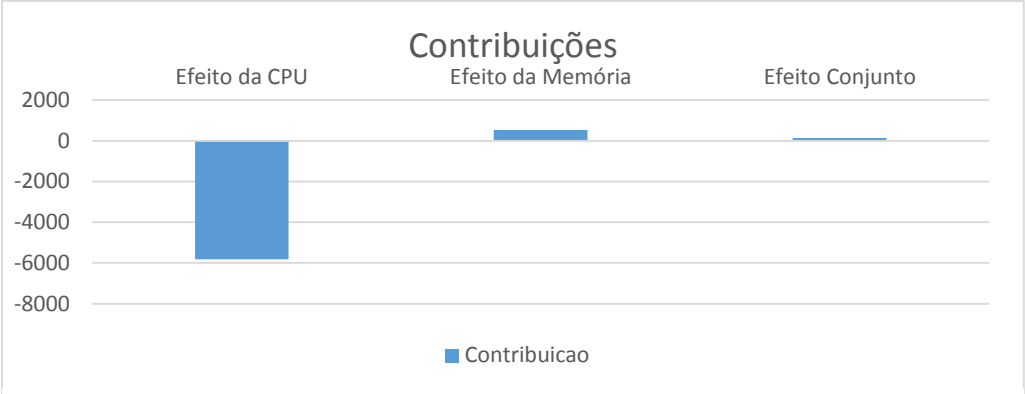

Figura 12 Contribuições absolutas CPU e Memoria Teste 1

Aplicando os valores das contribuições expostos na Eq 4.11, temos que a contribuição da CPU é de diminuição e representa 99% da contribuição, a contribuição da memória é de aumento do tempo, mas só representa 1% da contribuição total e a influência de um no outro não contribui com nada.

Os resultados do Teste 2 estão no Anexo D Resultados do teste 2 (Teste de memória) e estão resumidos na Tabela 3

				Média		Desvio Padrão
	CPU	Mem	Versão	Tempo de resposta (ms)		Tempo de resposta (ms)
1	ARM	256MB	2.3.3	216.012	±	4.373
2	ARM	256MB	4.0.3	72.452	±	2.165
3	ARM	512MB	2.3.3	224.491	±	5.196
4	ARM	512MB	4.0.3	56.370	±	1.411
5	x86	256MB	2.3.3			
6	x86	256MB	4.0.3	241.167	±	5.228
7	x86	512MB	2.3.3			
8	x86	512MB	4.0.3	208.458	±	4.510

Tabela 3 Resumo resultados teste 2

Aplicando a mesma estratégia do teste 1, temos, para a arquitetura ARM fixa:

$$y = 142.331 - 1.900 * q_{MEM} - 77.920 * q_{VER} - 6.140 * q_{MEM} * q_{VER} \qquad \text{Eq 4.12}$$

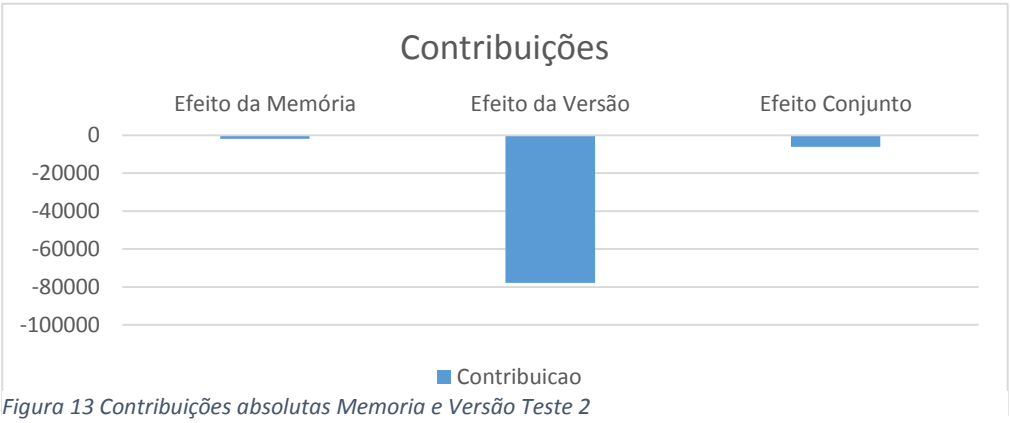

Figura 13 Contribuições absolutas Memoria e Versão Teste 2

Aplicando os valores das contribuições expostos na Eq 4.12, temos que a contribuição da memória é nula, a contribuição versão do Android é de diminuição do tempo e representa 99% e a influência de um no outro contribui também na diminuição, mas com apenas 1%.

Em seguida, mantendo a versão do Android na Ice Cream Sandwich, e variando o tamanho da memória e a arquitetura do processador, temos:

$$y = 144.611 + 80.200 * q_{CPU} - 12.197 * q_{MEM} - 4.156 * q_{CPU} * q_{MEM} \quad \text{Eq 4.13}$$

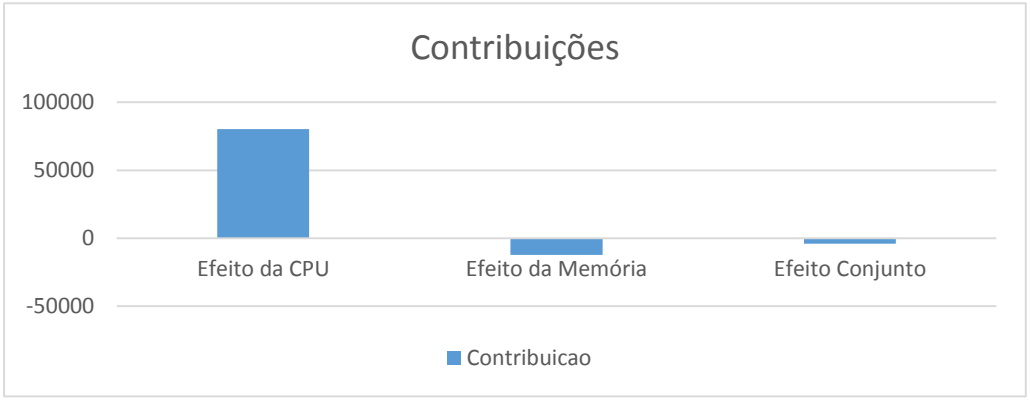

Figura 14 Contribuições absolutas CPU e Memória Teste 2

Aplicando os valores das contribuições expostos na, temos que a contribuição da CPU é de aumento do tempo e representa 97% da contribuição total. Já a contribuição da memória é de 2% na diminuição do tempo e a influência de um não contribui com nada.

4.2 Análise dos dados

De acordo com os resultados encontrados, podemos ver que para ambos os algoritmos testados, o efeito da versão Ice Cream Sandwich aparenta ser sempre

positivo em relação ao Gingerbread. Novos sistemas de gerenciamento de memória, códigos otimizados e correções de bugs podem ter influenciado para que isso acontecesse. Mas o que é mais importante é que para os algoritmos testados, podemos garantir que a versão mais atualizada é melhor do que a mais antiga.

Já para o quesito arquitetura de CPU, não houve uma unanimidade. Para o algoritmo de Lucas Lehmer, a melhora foi bastante representativa na utilização do processador com arquitetura Atom x86. Já para o algoritmo de teste de memória, o processador ARM mostrou-se mais capaz. Este experimento mostrou a importância de se testar as variações de configurações com o seu algoritmo ou com algum algoritmo que tenha o mesmo efeito da aplicação que se deseja utilizar no dispositivo final.

O efeito da memória foi nulo em ambos os testes. Os gráficos da Figura 15 e Figura 16 mostram o sutil efeito causado pela variação da memória no teste 1 e os da Figura 17 e Figura 18 o mesmo sutil efeito no teste 2. Nesses gráficos é possível perceber também a diferença do efeito do processador no teste 1 e no teste 2.

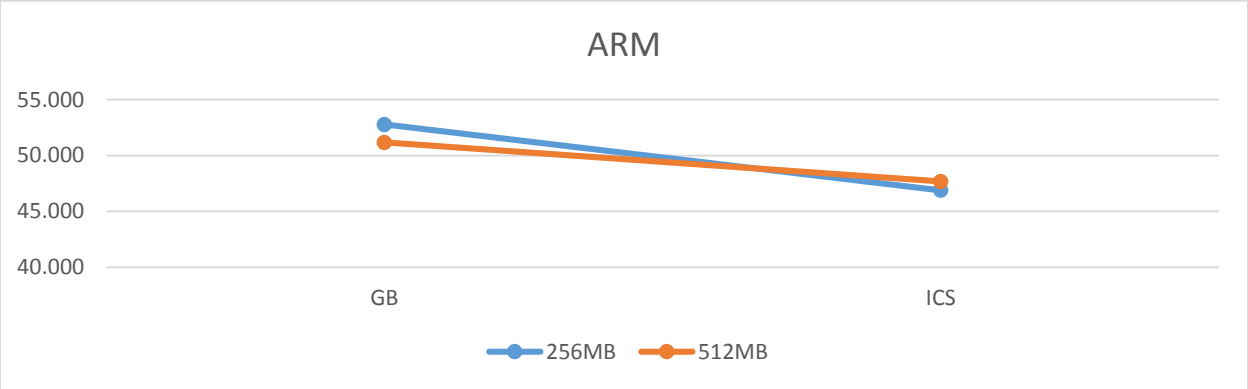

Figura 15 Efeito da memória para ARM fixo no teste1

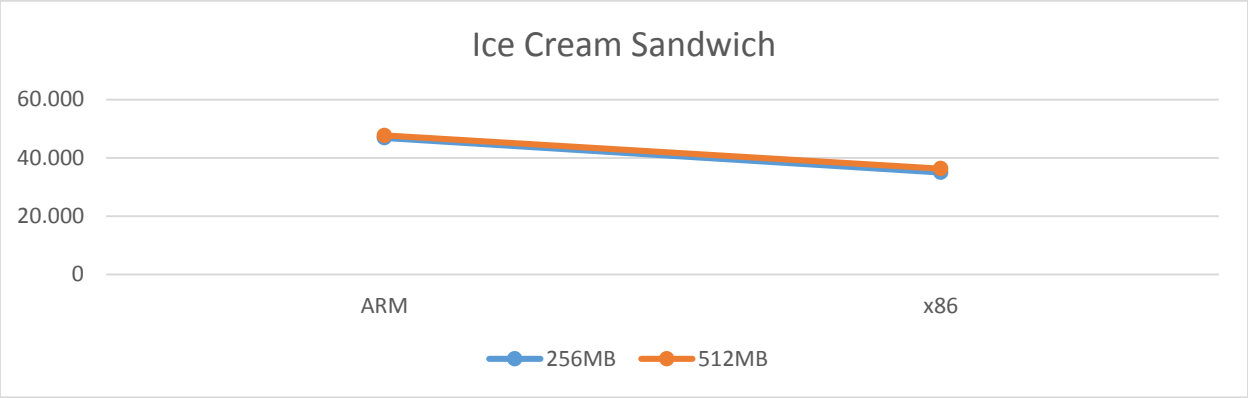

Figura 16 Efeito da memória para Ice Cream Sandwich fixo no teste 1

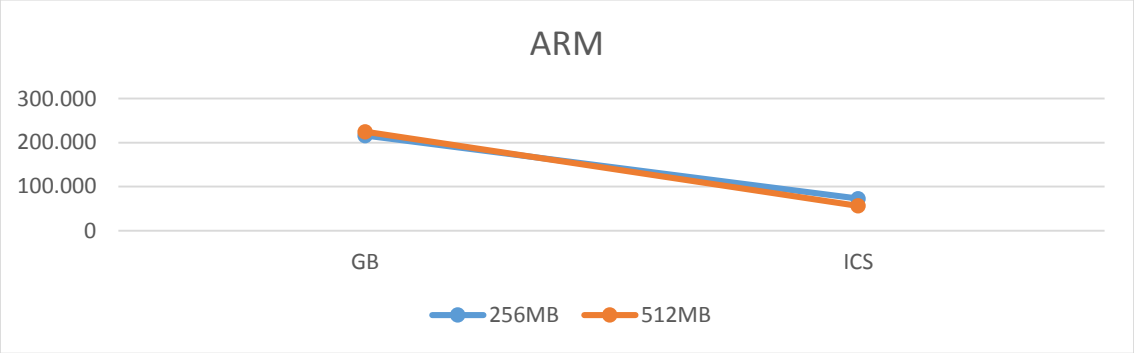

Figura 17 Efeito da memória para ARM fixo no teste1

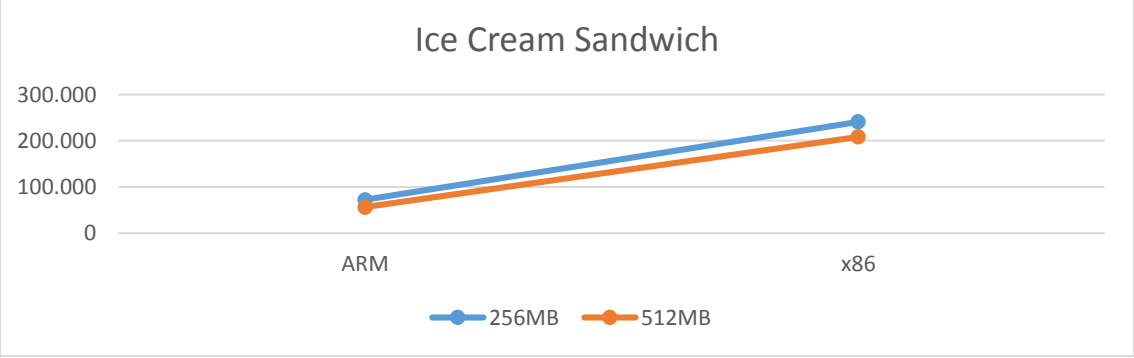

Figura 18 Efeito da memória para Ice Cream Sandwich fixo no teste 2

5 Conclusão

Estre trabalho mostrou quão importante e simples é realizar simulações e testes para a plataforma Android. Ele se propôs a facilitar as atividades da indústria de base tecnológica na hora da tomada de decisão por qual dispositivo escolher. No começo foi explicado tudo a respeito da arquitetura do sistema Android, com conceitos básicos e detalhes importantes para se ter em mente na hora de desenvolver para Android e analisar o desempenho dos testes. Foi também detalhado como utilizar as ferramentas necessárias para se desenvolver e testar aplicativos Android.

Posteriormente, foi explicado como se analisar os dados obtidos, com metodologias bem estabelecidas. O projeto de experimento fatorial 2^k não foi só detalhado, mas também analisado e aplicado.

Em trabalhos futuros, deve-se existir o empenho para melhorar a efetividade da intenção deste. Novas simulações devem ser feitas e novas metodologias de análise de desempenho explicadas e exemplificadas.

Referências

Nauman, M., & Khan, S. (Marco de 2010). Design and Implementation of a Fine-grained. *International Journal of Advanced Information Technology*.

"What is Android?". (2013). Fonte: Site Android: http://developer.android.com/guide/basics/what-is-android.html

AnTuTu Benchmark no Google Play. (2013). Fonte: https://play.google.com/store/apps/details?id=com.antutu.ABenchMar

ARMv7. (2013). Fonte: http://www.arm.com/products/processors/cortex-a/index.php

Barros Neto, B. (1995). *Planejamento e otimização de experimentos.* Campinas: Editora da Unicamp.

Cohen, H. a. (1987). Implementation of a new primality test. *Mathematics of Computation*, 48, 103-121.

Cryer, J. D., & Miller, R. B. (1994). *Statistics for Business 2: Data Analysis and Modeling.* Duxbury.

Dashboard Android. (Abril de 2013). Fonte: http://developer.android.com/about/dashboards/index.html

Google Play. (2013). Fonte: http://play.google.com

Intel Atom. (2013). Fonte: http://www.intel.com/pt_BR/consumer/products/processors/atom-family.htm

J.W.Bruce. (1993). A Really Trivial Proof of the Lucas-Lehmer Test. *The Americas Mathematical Monthly*, 100 (4): 370-371.

Jain, R. (1991). *Art of Computer Systems Performance Analysis Techniques For Experimental Design.* John Wiley & Sons, INC.

Macario, G., Torchiano, M., & Violante, M. (2009). An in-vehicle infotainment software architecture based on Google Android. *SIES - Swiss International Entrepreneruship Survey*, pp. 257-260.

Maia, C., Nogueira, L. M., & Pinho, L. (Julho de 2013). Evaluating Android OS for Embedded Real-Time Systems. *6th International Workshop on Operating Systems Platforms for Embedded Real-Time Applications*, pp. 63-70.

Montgomery, D. C. (2001). *Design and Analysis of Experiments* (5th ed.). John Wiley & Sons, INC.

Quadrant Standard no Google Play. (2013). Fonte: https://play.google.com/store/apps/details?id=com.aurorasoftworks.quadrant.ui.standard

Shabtai, A., Fledel, Y., Kanonov, U., & Elo, Y. (2009). Google Android: A State-of-the-Art Review of Security Mechanisms. *The Computing Research Repository*, p. CoRR abs/0912.5101.

Site Android. (2013). Fonte: http://www.android.com

Site Eclipse. (2013). Fonte: http://www.eclipse.org

Site Miniand. (2013). Fonte: http://www.miniand.com

Site Prime95. (04 de 2013). Fonte: http://www.mersenne.org/freesoft/

Vellamo no Google Play. (2013). Fonte: https://play.google.com/store/apps/details?id=com.quicinc.vellamo

Anexo A – Versões do Android

API	Codinome	Versão	Distribuição
1	-	1.0	0,0%
2	-	1.1	0,0%
3	Cupcake	1.5	0,0%
4	Donut	1.6	0,1%
5	Eclair	2.0	0,0%
6	Eclair	2.0.1	0,0%
7	Eclair	2.1	1,7%
8	Froyo	2.2	4,0%
9	Gingerbread	2.3-2.3.2	0,1%
10	Gingerbread	2.3.3-2.3.7	39,7%
11	Honeycomb	3.0	0,0%
12	Honeycomb	3.1	0,0%
13	Honeycomb	3.2	0,2%
14	Ice Cream Sanwich	4.0-4.0.2	0,0%
15	Ice Cream Sanwich	4.0.3	29,3%
16	Jelly Bean	4.1	23,0%
17	Jelly Bean	4.2	2,0%

Tabela 4 - versões do Android

Anexo B Código dos testes

```
public class Testes extends Activity {
	@SuppressLint("newApi")

	static int mega = 8*1024*1024;		// 8MB
	static int iter = 40;			// número de repetições
	static int option = 2;			// opcao de teste 1: Lucas Lehmer 2:Memoria

	//teste de primalidade
	public static boolean isPrime(int p) {
		if (p == 2)
			return true;
		else if (p <= 1 || p % 2 == 0)
			return false;
		else {
			int to = (int)Math.sqrt(p);
			for (int i = 3; i <= to; i += 2)
				if (p % i == 0)
					return false;
			return true;
		}
	}

	//teste para saber se um dado numero é primo de Mersenne
	public static boolean isMersennePrime(int p) {
		if (p == 2)
			return true;
		else {
			BigInteger m_p = BigInteger.ONE.shiftLeft(p).subtract(BigInteger.ONE);
			BigInteger s = BigInteger.valueOf(4);
			for (int i = 3; i <= p; i++)
				s = s.multiply(s).subtract(BigInteger.valueOf(2)).mod(m_p);
			return s.equals(BigInteger.ZERO);
		}
	}

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);

		Intent intent = getIntent();
		String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);

		TextView textView = new TextView(this);
		textView.setTextSize(40);


			switch(option){
			case 1:
				TimingLogger timings = new TimingLogger("LucasL", "Teste de
Lucas-Lehmer");
				for (int it = 0; it < iter; it++ ){
					for (int p = 3; p <= 1000; p += 2){
						if (isPrime(p) && isMersennePrime(p)){
							message+= ".";
						}
					}
					timings.addSplit("."+it);
					message+= " ";
				}
				timings.dumpToLog();

			case 2:
				TimingLogger timings2 = new TimingLogger("MemTest", "Teste de
memoria");
				for (int it = 0; it < iter; it++ ){
					byte [] arrayint = new byte[mega];
					System.out.println("teste de uso de memoria");

					for ( int i=0; i < mega; i++ )
					{
						arrayint[i]  = (byte) ( Math.random() * 128 );
```

```
                                        }

                                        for ( int i=0; i < mega; i++ )
                                        {
                                                int j =  arrayint[i];
                                                arrayint[i] = (byte)(j & 0x0F);
                                        }

                                        for ( int i=0; i < mega; i++ )
                                        {
                                                int j =  (byte) ( Math.random() * mega );
                                                int k =  (byte) ( Math.random() * mega );
                                                j = Math.abs(j);
                                                k = Math.abs(k);
                                                arrayint[i] = (byte) (arrayint[j] +
arrayint[k]);
                                        }

                                        message+=" array =>" + arrayint[mega/2] + " ";
                                        timings2.addSplit("."+it);
                                }
                                timings2.dumpToLog();
                        }



                textView.setText(message);

                //set the text view as activity layout
                setContentView(textView);
        }

        @Override
        public void onDestroy(){
                super.onDestroy();

                android.os.Debug.stopMethodTracing();
        }

        public boolean isExternalStorageWritable() {
                String state = Environment.getExternalStorageState();
                if (Environment.MEDIA_MOUNTED.equals(state)) {
                        return true;
                }
                return false;
        }
}
```

Anexo C Resultados do Teste 1 (Lucas Lehmer)

	ARM 256MB GB	ARM 512MB GB	ARM 256MB ICS	ARM 512MB ICS	x86 256MB ICS	x86 512MB ICS
1	53.023	51.557	50.825	48.736	35.473	42.382
2	56.618	50.899	47.434	47.699	34.959	35.545
3	59.930	50.667	47.089	47.357	34.531	35.757
4	50.369	51.029	47.353	48.560	34.735	35.550
5	49.581	51.020	48.280	48.899	34.299	36.717
6	50.326	51.519	50.123	47.978	34.498	38.374
7	52.263	51.186	47.528	47.624	34.414	37.175
8	56.398	51.260	48.277	47.291	34.543	35.993
9	56.228	51.194	47.445	47.951	34.708	35.489
10	59.629	50.001	49.388	47.493	34.642	35.747
11	58.841	50.646	47.108	49.388	34.871	37.111
12	58.439	49.802	47.535	47.957	34.284	39.062
13	59.375	49.549	47.216	49.732	34.538	34.869
14	53.745	50.384	47.476	48.815	34.476	35.040
15	50.742	49.960	47.297	47.953	34.572	34.961
16	51.389	50.012	46.584	49.064	34.399	34.982
17	51.443	50.337	45.980	48.842	34.810	35.628
18	49.508	50.707	44.776	47.616	34.637	39.106
19	50.640	51.037	44.843	46.515	34.351	37.093
20	50.044	50.063	44.541	47.127	34.535	37.276
21	49.830	49.522	45.034	47.955	34.450	36.763
22	54.475	62.207	45.525	47.774	34.711	37.296
23	51.984	49.157	45.668	47.339	35.115	37.021
24	53.178	52.623	45.251	47.804	35.483	37.330
25	48.657	51.574	47.908	48.032	35.476	36.510
26	50.894	51.737	53.128	47.684	35.200	35.279
27	50.846	53.123	52.002	49.044	35.454	35.245
28	48.536	50.872	50.769	47.361	35.573	36.368
29	49.643	51.905	50.033	47.761	35.556	39.507
30	51.426	52.311	44.985	46.507	35.620	35.121
31	56.847	53.131	46.341	46.571	35.561	35.189
32	53.695	50.614	45.846	46.024	35.578	35.233
33	57.644	50.276	45.036	47.714	35.503	34.960
34	54.797	50.465	45.858	47.314	35.291	35.130
35	52.268	50.143	44.770	46.545	35.520	35.359
36	51.219	50.174	44.546	46.921	35.658	34.795
37	50.542	52.952	44.059	47.084	35.304	34.596
38	48.603	50.237	43.951	46.703	35.448	34.944
39	48.634	50.754	43.675	47.108	35.683	36.190
40	48.732	50.189	44.191	45.318	35.377	36.022

Tabela 5 Resultados teste 1 (Lucas Lehmer)

Anexo D Resultados do teste 2 (Teste de memória)

	ARM 256MB GB	ARM 512MB GB	ARM 256MB ICS	ARM 512MB ICS	x86 256MB ICS	x86 512MB ICS
1	216.162	77.020	214.302	58.525	270.523	209.040
2	215.175	73.782	214.585	54.910	241.754	202.291
3	210.376	71.531	213.702	57.964	240.257	205.479
4	217.242	72.231	211.931	58.769	239.076	203.406
5	214.503	70.577	232.987	58.586	239.409	210.208
6	216.044	69.262	223.971	57.505	240.327	211.515
7	217.675	69.163	221.158	54.946	244.253	209.769
8	220.161	71.874	219.521	56.962	241.408	222.267
9	208.314	77.535	225.507	55.329	237.206	219.456
10	222.624	74.052	226.813	55.172	242.702	212.916
11	211.576	73.178	230.828	56.487	237.600	208.224
12	213.163	73.341	226.549	58.250	243.337	203.922
13	218.009	69.217	226.112	57.328	240.767	205.203
14	222.629	71.504	226.558	58.266	243.106	202.976
15	213.716	69.805	226.398	55.877	242.274	206.681
16	217.221	69.585	231.577	57.712	239.610	204.872
17	212.322	73.878	226.285	57.565	242.543	204.520
18	223.047	74.543	218.196	57.857	239.238	211.452
19	215.338	75.223	225.340	54.835	237.890	211.231
20	215.013	72.217	227.530	56.912	238.059	210.791
21	214.850	73.947	231.219	54.357	237.454	205.211
22	219.521	69.330	223.952	56.313	236.600	209.232
23	212.788	75.224	229.533	57.098	244.131	204.196
24	220.278	75.518	227.203	57.537	243.250	204.902
25	223.020	73.267	218.454	54.885	242.047	204.504
26	221.502	73.406	222.417	58.408	237.475	213.506
27	212.165	72.469	220.433	54.214	242.684	214.621
28	214.474	72.304	229.534	55.771	240.793	214.835
29	213.159	75.360	224.614	55.554	239.598	207.749
30	216.677	70.413	229.207	57.714	238.418	205.857
31	210.402	72.845	222.011	56.663	240.276	208.066
32	214.096	73.747	229.698	54.405	239.149	207.865
33	208.590	71.355	229.134	55.800	237.858	205.566
34	221.189	70.238	222.591	54.567	243.668	206.359
35	208.603	69.671	225.907	55.435	241.278	206.309
36	222.856	71.346	219.915	54.816	240.401	211.592
37	210.439	71.020	225.638	54.790	243.181	202.095
38	222.489	73.418	221.949	55.429	238.416	211.798
39	216.878	71.980	229.658	55.628	238.787	206.568
40	216.190	71.689	226.735	55.656	239.877	211.281

Tabela 6 Resultados Teste 2 (Memória)