
Universidade Federal de Pernambuco
Centro de Ínfomática

Graduação em Engenharia da computação

# Avaliação de desempenho de memórias flash em dispositivos móveis.

Newton Leal Barbosa

Trabalho de Graduação

Recife
4 de dezembro de 2018

Universidade Federal de Pernambuco
Centro de Ínfomática

Newton Leal Barbosa

# Avaliação de desempenho de memórias flash em dispositivos móveis.

*Trabalho apresentado ao Programa de Graduação em Engenharia da computação do Centro de Ínfomática da Universidade Federal de Pernambuco como requisito parcial para obteno do grau de Bacharel em Engenharia da computação.*

Orientador: *Prof. Dr. Eduardo Antônio Guimarães Tavares*

Recife
4 de dezembro de 2018

*Dedico a todos que desejam fazer o mundo um lugar melhor.*

# Agradecimentos

Primeiramente, eu gostaria de agradecer a minha família, e com certeza não estaria aqui sem esse apoio. Em especial aos meus pais, que me ensinaram o real valor de uma educação de qualidade e me dar o suporte necessário, profissional e pessoal. Gostaria de agradecer também aos meus amigos, que me ajudaram a manter a sanidade mental durante essa graduação com conversas baratas e risadas relaxantes. Aos irmãos Carlos (Ikinho e Júnio) por me ajudarem na apresentação e em situações de completo desespero. Aos meus amigos da universidade que estiveram comigo durante toda essa trajetória, em especial a Alice, Gabriel, Mayara, Marina, Ihago, Gabriel Bandeira, Djeefther, Titi e Yuri.

Também agradeço a Natália Schardosim por ter me apoiado, me incentivado e ficado ao meu lado durante a realização deste trabalho. E por ter sido a surpresa mais maravilhosa que me aconteceu em muito tempo.

Também agradeço o apoio ao convênio CIn/Motorola que disponibilizou dispositivos e ferramentas para os experimentos realizados neste trabalho.

Finalmente, agradeço ao professor Eduardo Tavares que me orientou ao longo deste trabalho e tornou possível a realização do mesmo.

*Just leave me alone, I know what I'm doing.*
—KIMI RÄIKKÖNEN (2012)

# Resumo

Dispositivos móveis são uma parte muito importante no dia-a-dia da maioria da população mundial. São os grandes responsáveis por uma mudança nas relações sociais que temos hoje em dia. Com o crescimento de dados trafegados em dispositivos móveis, várias tecnologias de armazenamento foram implementadas para suprir os requisitos dos mesmos. Este trabalho faz uma avaliação de desempenho entre os diferentes sistemas mais utilizados hoje em dia através de uma série de operações de arquivos em dispositivos *Android*.

Foi desenvolvida uma ferramenta capaz de ler/escrever/remover arquivos de/para um celular *Android* e extrair métricas de desempenho destas operações. Foram feitos experimentos com memória eMMC, UFS e SD Card.

Os resultados obtidos mostraram que eMMC ainda possui um desempenho satisfatório na maioria dos casos, até melhores que o UFS em operações envolvendo arquivos pequenos (4kb) enquanto que UFS teve uma performance melhor quando foram feitos experimentos com arquivos maiores (1024kb).

**Palavras-chave:** Memória Flash, eMMC, UFS, Memória não-volátil, Android, SD Card, Dispositivos Móveis

# Abstract

Mobile devices are an important part of the people's lives nowadays. They are one of the major responsible for a change in our social relationships and how we interact with each other. With the growth of amount of data consumed worldwide, several mobile storage technologies were developed to conform with the mobile devices requirements. This article makes an performance evaluation between the major systems used in mobile devices throughout a series of file operations in *Android* devices.

A tool was developed capable of read/write/delete files from/to an *Android* and extract performance metrics of those operations. The experiments were made with eMMC, UFS and SD card memory. The results obtained showed that eMMC still has a satisfactory performance in most cases, sometimes even better than UFS in operations involving small files (4kb). Whilst UFS has a better performance when experiments were made with bigger files (1024kb).

**Keywords:** Flash Memory, eMMC, UFS, Non-volatile Memory, Android, SD Card, Mobile Devices

# Sumário

# Lista de Figuras

# Lista de Tabelas

CAPÍTULO 1

# Introdução

Os Dispositivos móveis são cada vez mais utilizados por toda a população, o que significa que cada vez mais informação deve ser armazenada nestes dispositivos. Por isso, houve a necessidade do desenvolvimento de novos sistemas capazes de arquivar essa nova quantidade de informações, enquanto cumpriam com requisitos específicos de dispositivos móveis como economia de energia, rapidez, leveza e resistência. [CAB+15].

Com isso, os sistemas de armazenamento móvel estão em constante evolução e fabricantes estão buscando novas maneiras de otimizar esta parte do dispositivo.

O sistema*Android* sofreu a primeira revisão em 2008, em desde então, está em contínuo crescimento de participação do mercado, até que finalmente, se tornou o sistema operacional mais utilizado no mundo.[Mob]

De 2008 pra cá, os dispositivos móveis mudaram, de fazerem ligações e mandarem curtas mensagens de texto, e agora são capazes de se conectar à internet de maneira super-rápida, reproduzir vídeos de qualidade 4K, jogos, ou até servirem como um substituto do próprio computador (Samsung DeX). [Sam]

A distribuição do *Android* em 2010, do Nexus S quando foi lançado, precisava de aproximadamente 100MB para ser instalada, enquanto que no celular mais recente da Google, o Pixel 3 o espaço utilizado pelo sistema chega a 9.6GB. Esse aumento da complexidade do sistema *Android* e as demandas de UX (User eXperience) fizeram com que todos os periféricos e o hardware fossem capazes de suprir as necessidades do OS.

Com o sistema de armazenamento interno não foi diferente, o que antes era apenas conjuntos de memória NAND com capacidade de alguns MB ou talvez 1GB nos melhores casos, se tornou um sistema complexo com densidade muito maior, melhorias na eficiência energética, interações complexas com o sistema e ainda assim, tudo isso é feito com um custo por bit relativamente baixo.

Para "aguentar"todas essas funcionalidades requeridas pelos usuários, o hardware do dispositivo foi evoluindo ao longo dos anos também.

Em contrapartida, as fabricantes de dispositivos móveis vivem uma concorrência implacável entre elas, então toda vantagem em relação a custo e/ou melhor UX em relação ao concorrente conta.

MMC (MultiMediaCard) é um dos sistemas mais antigos e consolidados que existem no mercado. Criado em 1997 pela Siemens e pela SanDisk, consiste em uma interface serial ligada a uma pilha de chips de memória. Originalmente foi utilizado memória Flash NOR porém foi mudado para memória do tipo NAND, que é o padrão até hoje pois, entre os benefícios comparado a NOR, memória flash NAND tem tempo de escrita e remoção menores, interface mais simples e uma densidade de dados maior. Portanto, memória NOR é mais adequada para

armazenamento de código e aplicações execute-in-place, enquanto que memória flash NAND é uma melhor candidata para armazenamento de dados.

O SD Card, é uma extensão do MMC, adicionado uma camada segura para proteção de dados com copyright (exigência da industria de multimídia) e em vez de usar um único pino para transferência serial de dados, utiliza-se de 4 pinos para criar uma interface serial que pode transferir até 8 bits de uma vez, permitindo uma velocidade de transmissão maior do que o MMC originalmente. [GA+][AP+11]

eMMC (embedded MutiMediaCard) é o sistema de armazenamento mais utilizado hoje em dia nos dispositivos móveis. Feito especialmente para os mesmos, consiste em um dispositivo embarcado com um microcontrolador e memória NAND. Ele interage com dispositivo móvel por meio de uma interface padrão que define vários parâmetros, tais como partição, inicialização pela partição de boot, proteção de escrita, consumo de energia entre outros. [Sta15]

Por outro lado, um sistema mais recente foi lançado e vem se tornando popular entre as fabricantes de dispositivos móveis. Este sistema é o UFS (Universal Flash Storage). O UFS consegue velocidades de leitura e escrita sequenciais equivalentes aos SSDs de Desktops e Laptops enquanto mantém o baixo consumo de energia dos eMMC. [Sta11]

## 1.1 Objetivos

Este trabalho tem como objetivo um estudo comparativo entre os sistemas de memória não volátil de dispositivos móveis, mais especificamente entre eMMC, UFS e SD Card, analisando as métricas de desempenho mais importantes para a experiência de usuário que são tempo de leitura, escrita e remoção de dados com diferentes tamanhos de arquivos, o que permite ter insights sobre a evolução dos sistemas de armazenamentos móveis e também auxiliar na escolha de qual sistema adotar dependendo da demanda do fabricante ou usuário.

Dentre os objetivos específicos deste trabalho podemos listar:

- Construção de um pequeno programa que lê, escreve e remove um arquivo de tamanho específico fornecido pelo o usuário.
- Construção de uma ferramenta que inicializa celulares, configura ciclos de leitura/escrita/remoção, mostra métricas graficamente e armazena em arquivos para maior análise
- Verificação dos experimentos realizados para uma análise estatística dos ciclos aplicados nas diferentes tecnologias apresentadas neste trabalho.

## 1.2 Organização do trabalho

Este trabalho está organizado em 5 capítulos. No segundo capítulo, será detalhada a revisão bibliográfica dos diferentes sistemas de armazenamento móvel. No terceiro capítulo, será detalhado os experimentos realizados. No quarto capítulo será discutido os resultados obtidos. O quinto capítulo será reservado às referencias bibliográficas e a trabalhos futuros.

CAPÍTULO 2

# Referencial Teórico

A memória não-volátil serve basicamente para armazenar dados mesmo após uma queda de energia ou desligamento do aparelho. Neste capítulo faremos uma análise detalhada dos vários sistemas avaliados neste trabalho.

## 2.1 MultiMediaCard

MMC é uma mídia universal de baixo custo para comunicação e armazenamento. Foi desenvolvido para suprir um grande leque de aplicações, como smartphones, câmeras digitais, MP3 players, brinquedos eletrônicos, etc. O objetivo do desenvolvimento do MMC são alta performance em baixo custo. Funcionalidades adicionais incluem baixo consumo de energia e alta vazão de dados na interface do cartão de memória.

O principal objetivo de design do MMC é de prover um sistema de armazenamento de um custo muito baixo e implementado como um "cartão", com um controlador simples e uma interface fácil de implementar.

O sistema MMC contém ao menos dois componentes:

- O Cartão multimídia
- O controlador MMC

O controlador pode implementar toda a aplicação, enquanto em outros pode ser dividido em diferentes componentes, são eles:

**1** Adaptador de aplicação - Faz tarefas orientadas à aplicação

**2** Adaptador MultiMediaCard - Implementa a interface padrão para o cartão

O núcleo do MMC é o microcontrolador, que é o que rege todas as atividades do cartão. Uma pilha de chips de memória NAND é onde os dados de memória realmente estão guardados os dados. O microcontrolador se comunica com a pilha de memória NAND por meio de uma interface de memória. A interface de memória implementa o protocolo NAND para trocar informação com a pilha de memória. Cada chip de memória NAND tem sua própria especificação descrita pela própria fabricante. Então, normalmente um chip de uma fabricante requer uma interface de memória para o mesmo. O microcontrolador é acessado pelo hospedeiro pela interface de protocolo. O MMC tem o seu protocolo, assim como SD.

Esses 3 componentes (microcontrolador, interface de protocolo e interface de memória) compõem o que é chamado de controlador

Pela imagem 2.1, conseguimos ver esses componentes representados pelo controlador de interface do cartão, que é ligado com o mundo exterior, e a interface do núcleo de memória, que gerencia os chips de memória NAND.

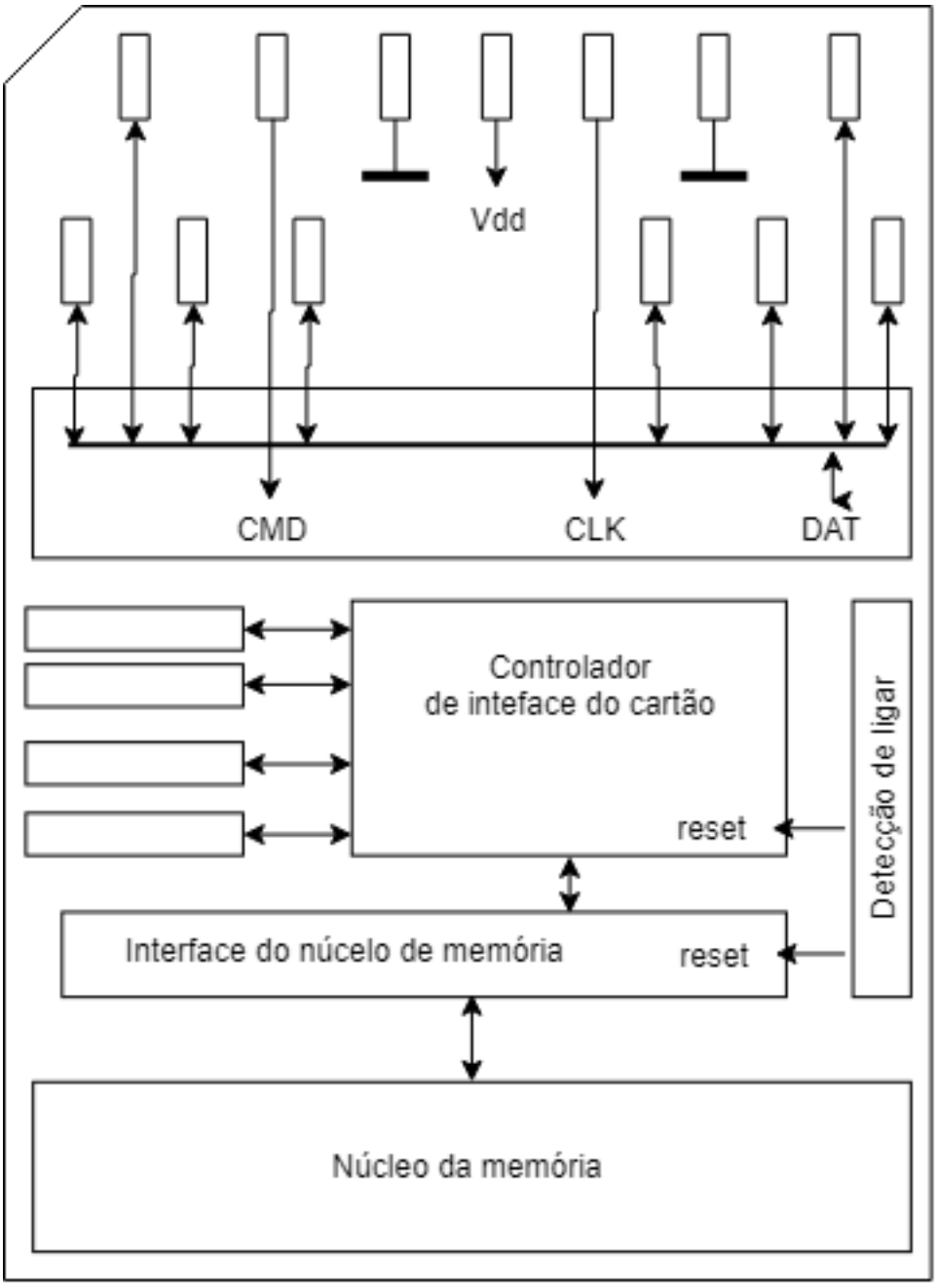


**Figura 2.1** Esquema gráfico da estrutura interna do MMC

## 2.2 Embedded MultiMediaCard

eMMC é um sistema embarcado complexo que consiste em chips de memória NAND, um controlador embarcado e um firmware (FW). O firmware tem o papel fundamental de implementar o acesso indireto de memória flash pela Flash Translation Layer (FTL)

Agora iremos falar um pouco sobre as especificações da eMMC, visto que é um dos focos do nosso trabalho.

### 2.2.1 Comandos

Comandos enviados para a memória eMMC são enviados por tokens. Os comandos tem tamanho de 48 bits e são enviados bit a bit pelo pino CMD:

| | Bit de começo | Bit de transmissão | Comando | Argumento | CRC7 | Bit de fim |
|---|---|---|---|---|---|---|
| Tamanho (bits) | 1 | 1 | 6 | 32 | 7 | 1 |

**Tabela 2.1** Representação de um token de comando para eMMC

O bit de começo, transmissão e fim são apenas para controle e definir os limites do comando. O índice de comando tem tamanho de 6 bits, permitindo um total de 64 comandos possíveis. Porém, mais comandos podem ser executados enviando mais comandos em sequência.

O campo de argumento tem um propósito diferente e depende do comando para a ser enviado. O CRC (Cycle Redundancy Check) é para verificar se o comando recebido é igual ao comando enviado, que pode acontecer algum erro devido à interferência ou erro no cartão. O CRC é calculado observando todo o token de comando recebido e é calculado a parte do processamento do comando. Se o cálculo for correto, o comando é executado, senão um erro é disparado.

Os comandos mais utilizados são os de leitura e escrita. O índice de comando para leitura um único bloco é 17, e para escrever um único bloco é 24. Nos dois casos, o campo de argumento é o endereço de memória a ser lido ou escrito. Note que por mais que o argumento só tenha espaço para 32 bits, pode ser endereçado mais do que 4GB, pois em memórias maiores que 4GB, o endereçamento é feito por setores.

Cada comando é respondido pelo cartão. É necessário para o hospedeiro saber se o comando foi recebido de maneira correta, se vai começar a transmissão de dados etc. O tipo de resposta mais comum tem tamanho de 48 bits e segue a seguinte estrutura:

| | Bit de começo | Bit de transmissão | Comando | Status do cartão | CRC7 | Bit de fim |
|---|---|---|---|---|---|---|
| Tamanho (bits) | 1 | 1 | 6 | 32 | 7 | 1 |

**Tabela 2.2** Representação de um token de resposta para eMMC

Os campos são muito parecidos com o de comando, porém há algumas diferenças. O comando é apenas o código do comando replicado. O Status do cartão é um amontoado de informação sobre o cartão, além do status retornando em relação ao comando.

Em um comando de leitura de um único bloco, o hospedeiro manda o comando pelo pino CMD, o cartão então responde se o comando foi aceito ou se ocorreu algum erro. Em seguida, a transmissão de dados começa pelo DAT.

O comando de escrever em um único bloco é bastante similar, neste caso, os dados são enviados do hospedeiro para o cartão. O hospedeiro primeiro espera a resposta do cartão se o comando foi aceito corretamente, e depois começa a transmissão dos dados. Após o término da transmissão, o cartão manda um sinal de ocupado para processar os dados recebidos do hospedeiro.

A memória é apagada por comandos de remoção (35, 36, 38), o primeiro comando, o hospedeiro indica qual é o endereço inicial e o comando 36 envia qual é o endereço final para determinar o intervalo a ser removido enquanto que no comando 38, o processo de remoção é efetivamente iniciado.

### 2.2.2 Flash Translation Layer

Flash translation layer é responsável por traduzir o endereço lógico do hospedeiro para o endereço físico da memória NAND. Esta camada é necessária para cumprir duas das características da tecnologia de memória flash NAND:

- Apagar antes de escrever
- Assimetria entre escrita/leitura (por página) e remoção (por bloco)

A FTL foi criada também para aliviar a quantidade de trabalho pelo sistema operacional, rotinas de software podem ser implementadas para emular memórias NAND Flash para editar páginas individualmente. A FTL é uma camada intermediária entre o sistema de arquivos e a mídia de armazenamento. Isto é o que permite o sistema operacional escrever no sistema de memória sem se preocupar com a implementação da eMMC.

Como não é possível atualizar dados no mesmo lugar, uma atualização de dados consiste em uma escrita do dado atualizado e a invalidação na FTL da versão anterior e finalmente o mapeamento lógico para o físico do novo local. Então, a principal função da FTL é manter em tabelas além da tradução de endereço lógico pro endereço físico, é de guardar quais páginas estão inválidas e quais estão disponíveis para escrita.

Neste exemplo da figura 2.2, a página 0 foi escrita no endereço n, em seguida a página 1 foi escrita no endereço n+1. A página 0 foi então atualizada, e em vez de atualizar no mesmo endereço, é escrita uma nova versão da página 0 no endereço n+2 e então é invalidada a página 0 no endereço n. E assim sucessivamente.

Quando o sistema tem muitas páginas inválidas, um programa chamado Garbage Collector (GC) é ativado, que basicamente copia todas as páginas válidas de um conjunto de blocos selecionados para um outro conjunto de blocos, e então é liberado espaço em todo o bloco.[ACCS15].

O garbage collector é uma parte crítica para o funcionamento do sistema de memória, um dos pontos a ser examinado é a eficiência do garbage collector. A eficiência do garbage collector é definida pela razão entre o número de páginas invalidas no bloco e o número total de

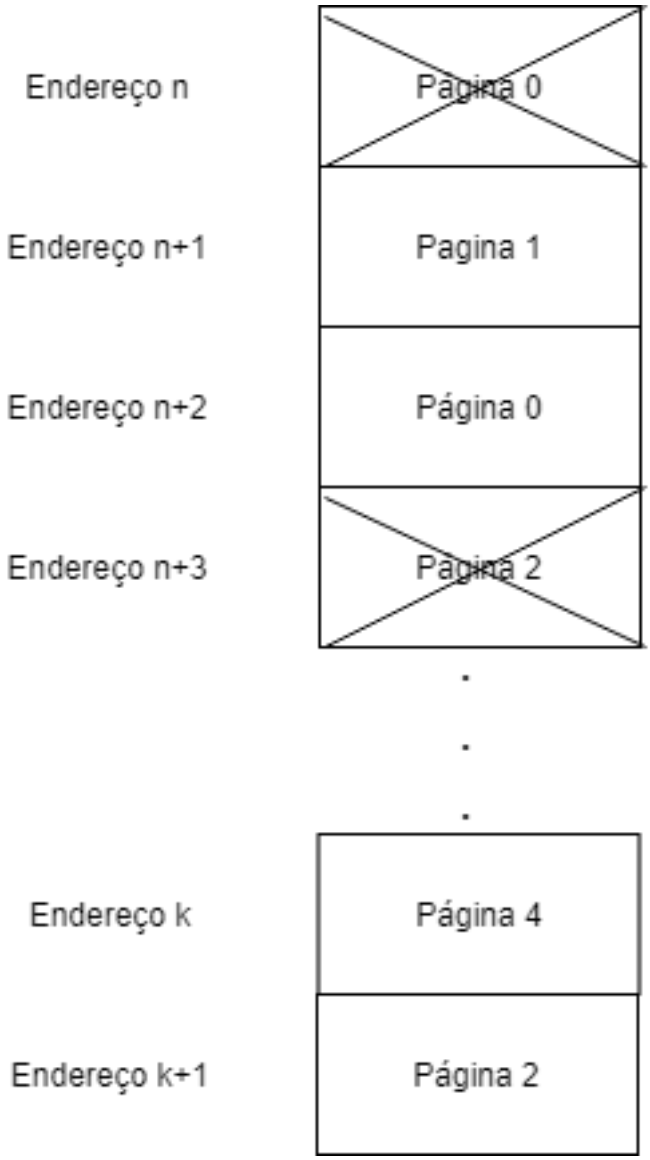


**Figura 2.2** Mapeamento de endereço lógico para físico

páginas nesse bloco. Quanto maior a quantidade de páginas invalidas num bloco, maior a eficiência do garbage collector. Os benefícios de ter várias páginas inválidas antes de disparar o garbage collector são dois:

- Com poucos endereços válidos (a maioria estão inválidos), poucas páginas válidas serão copiadas para novos endereços.
- Depois da execução do garbage collector, uma maior quantidade de dados será liberada para escrita

O momento que o garbage collector deve ser excetuado é assunto de um certo debate, pois

sistemas que exigem uma alta escrita/leitura de dados requer uma grande quantidade de espaço, e caso durante a execução do programa não tenha mais espaço, o programa terá que parar para executar o garbage collector. Então, pode-se achar que é melhor executar o garbage collector o mais frequente possível. Porém, ao executar numa frequência muito alta, a eficiência do garbage collector diminui, portanto é necessário encontrar um balanceamento ideal ao decidir a frequência de execução do GC.

Blocos de memória flash tem um número limitado de operações possíveis antes que pare de funcionar. Um dos papéis da FTL é ter um algoritmo de nivelamento de desgaste da memória. Nivelamento de desgaste é útil em qualquer sistema que escreve informação num sistema flash. Se o dispositivo não é capaz de identificar o nível de desgaste, um bloco pode ser usado exclusivamente ou com uma frequência muito mais alta do que os outros pois tem uma informação muito utilizada. Isto pode causar um desgaste na memória flash e finalmente inutilizar este bloco, causando perda de espaço de armazenamento ou até perda total de informação. Dispositivos mais modernos conseguem guardar estatísticas sobre o número de vezes que cada bloco é apagado e/ou escrito e move os dados de acordo para que todos os blocos tenham um nível de desgaste parecido. Normalmente o algoritmo de desgaste de nivelamento vai garantir que o número de remoções do bloco que foi apagado mais vezes e o que foi apagado a menor quantidade de vezes fiquem perto um do outro. Isto aumenta a quantidade de tempo que a memória flash chega ao fim do seu ciclo de acordo com a especificação do fabricante. [BG11]

## 2.3 Secure Digital Card

O padrão SD foi introduzido em Agosto de 1999, foi desenvolvido também como uma extensão do MMC (assim como o eMMC) que era usado na época. O formato SD inclui quatro famílias de cartões em diferentes formatos. As quatro famílias são o original SD Capacidade padrão (Standard capacity) (SDSC), o Alta Capacidade (High-Capacity)(SDHC) e Capacidade eXtendida (SDXC). Em relação à forma, existem 3 padrões que são o tamanho tradicional, tamanho mini e tamanho micro.

A maioria dos tocadores de música portátil começaram usando MMC como sistema de armazenamento, porém a indústria fonográfica não apoiava o uso de MMC porque sistemas usando MMC facilitaria a pirataria de música. Então, a Toshiba adicionou criptografia em hardware no padrão MMC e o renomeou como SD. Isto permitiu o gerenciamento de direitos digitais para a indústria musical.

### 2.3.1 Barramento SD

O barramento SD é a conexão entre o controlador do hospedeiro o cartão SD. A figura 2.3 Mostra um barramento SD com o controlador do hospedeiro e o cartão SD. Os sinais transmitidos no barramento SD incluem um pino de clock gerenciado pelo hospedeiro, um pino bidirecional para o CMD e de um a quatro para o DAT, como pode ser visto na figura 2.1. Os comandos recebidos e enviados pelo SD seguem a mesma forma do eMMC descrita na seção 2.2.1 pois como já foi dito, SD é apenas uma extensão do MMC. Portanto, também consiste em comandos de 48 bits no mesmo formato.

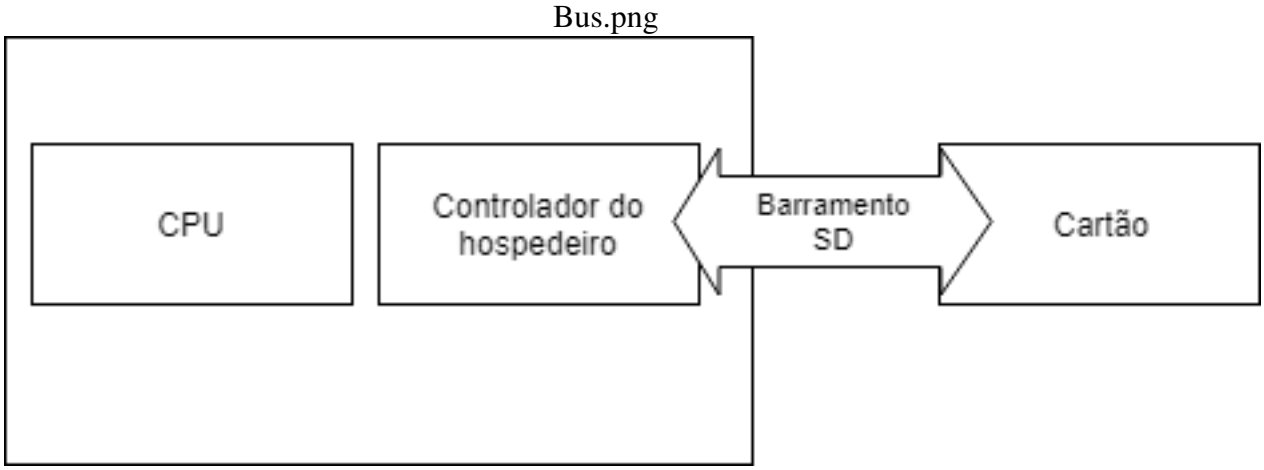


**Figura 2.3** Mapeamento de endereço lógico para físico

A especificação do barramento do SD define a interface do sinal físico no barramento SD, a definição de cada comando e respostas, um conjunto de registradores dentro do cartão SD, o estado interno do cartão e o status do cartão. A especificação do cartão SD também define a sequência de comandos para inicializar e permitir transferência de dados no barramento.

### 2.3.2 Controlador do hospedeiro

O controlador do hospedeiro é a parte do hardware que atua como a ponte entre o processador do hospedeiro e o barramento SD. O mesmo tem uma especificação definida de forma bastante estrita. A definição de um padrão permite que os hospedeiros utilizem cartões de diferentes fabricantes sem se preocupar com interfaces mais especificas.

Da perspectiva da CPU, o controlador do hospedeiro consiste num conjunto de registradores de 256 bytes que é mapeado na memória do sistema ou como uma E/S. Uma transação no barramento SD é iniciada pelo software lendo ou escrevendo nesse conjunto de registradores. Este conjunto de registradores é usado pelo software do driver do hospedeiro para operar o cartão. As operações incluem detecção, configuração, acesso e gerenciamento de energia do cartão. O controlador do hospedeiro, similarmente a outros controladores modernos, possui acesso direto à memória para fazer transferências sem precisar da intervenção da CPU.

### 2.3.3 Design do cartão SD

O design do cartão SD é bastante similar ao do MMC, portanto podemos ver pela figura 2.1. A "Detecção de ligar"é responsável pela lógica quando o cartão é ligado ou desligado. O controlador de interface de cartão é a unidade de processamento dentro do cartão. Existe também a memória para armazenamento e uma memória interna para uso do processador interno do cartão.

Para a implementação da memória SD, é utilizada memórias flash NAND, tal como as outras tecnologias que é gerenciada pelo controlador interno com memória interna para operações

internas. O controlador SD gerencia todas as funções a nível físico ou lógico como decodificação de comandos, geração de resposta, CRC, gerenciamento de status. Vários comandos podem ser processados pelo controlador SD sem nenhum envolvimento de outros componentes do cartão, Entretanto, o acesso à memória tem que ser encaminhado do controlador SD para outros módulos do cartão. Por exemplo: um comando de leitura ou escrita de um hospedeiro é encaminhado para o bloco de hardware que é responsável por se comunicar com a NAND flash.

O cartão SD também tem um gerenciamento de nivelamento de desgaste dos blocos de memória similar ao da eMMC, para manter a sanidade do cartão e também prolongar a sua vida útil.

## 2.4 Universal Flash Storage

UFS é o mais novo padrão de arquitetura de memória flash. É o padrão mais novo da JEDEC que provê a velocidade de leitura e escrita equivalente ao SSD porém com o baixo consumo de energia do eMMC. A intenção por trás do padrão UFS é ser uma única interface tanto para sistemas de armazenamento embarcados ou removível (cartões UFS). O padrão UFS utiliza-se de um conjunto de padrões já existentes, como o conjunto de comandos SCSI já utilizados por SSD, O M-PHY, que é um padrão de camada física de permite alta taxa de transmissão de dados, e UniPro. A UFS também mantem uma certa compatibilidade com eMMC para facilitar a adoção e o desenvolvimento. UFS utiliza SCSI para a camada superior M-PHY UniPro para a camada superior. M-PHY é responsável pela camada física e UniPro como a camada de enlace.

### 2.4.1 Arquitetura do sistema UFS

A arquitetura simplificada de um sistema que utiliza UFS é descrita na figura 2.4. A CPU se comunica com o dispositivo UFS através do controlador do hospedeiro UFS. Para a comunicação, é utilizado UniPro e M-PHY enquanto que o dispositivo UFS é implementado por cima dos comandos SCSI e operações.

### 2.4.2 Arquitetura de comunicação do UFS

O UFS tem uma arquitetura de comunicação baseada em camadas, que é baseado na arquitetura SCSI SAM-5. SAM-5 significa modelo de arquitetura SCSI 5 (SCSI Architeture Model 5), que define, num nível de abstração, como dispositivos SCSI devem se comunicar. A arquitetura de comunicação UFS consiste nas seguintes camadas:

- UCS (Conjunto de comandos UFS)
- UTP (Protocolo de transporte UFS)
- UIC (Interconexão UFS)

A camada de conjunto de comandos é a interface para o software. Ela utiliza do padrão SCSI como o (baseline) para a especificação UFS. A camada do protocolo de transporte é de

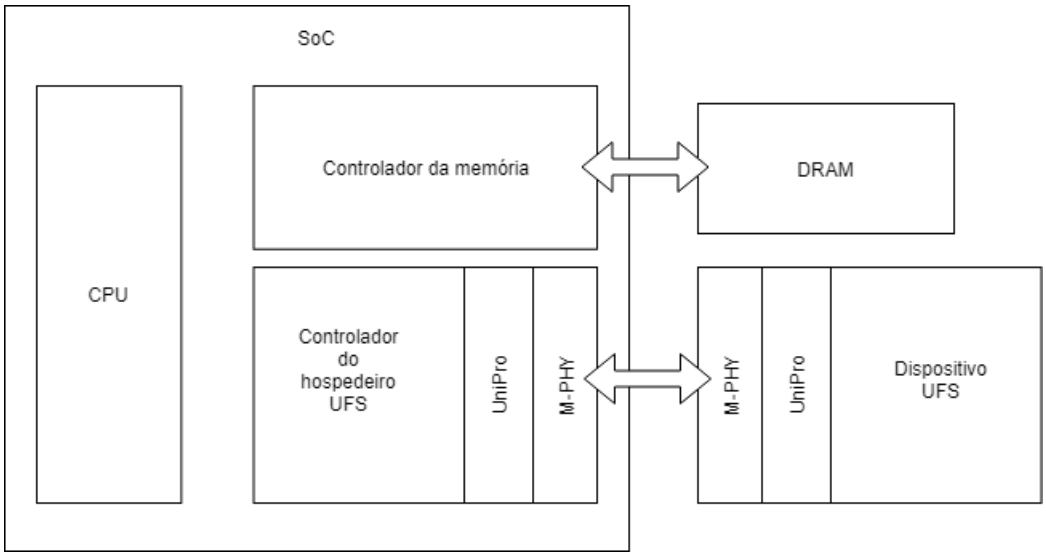


**Figura 2.4** Arquitetura de um sistema em alto nível que utiliza UFS

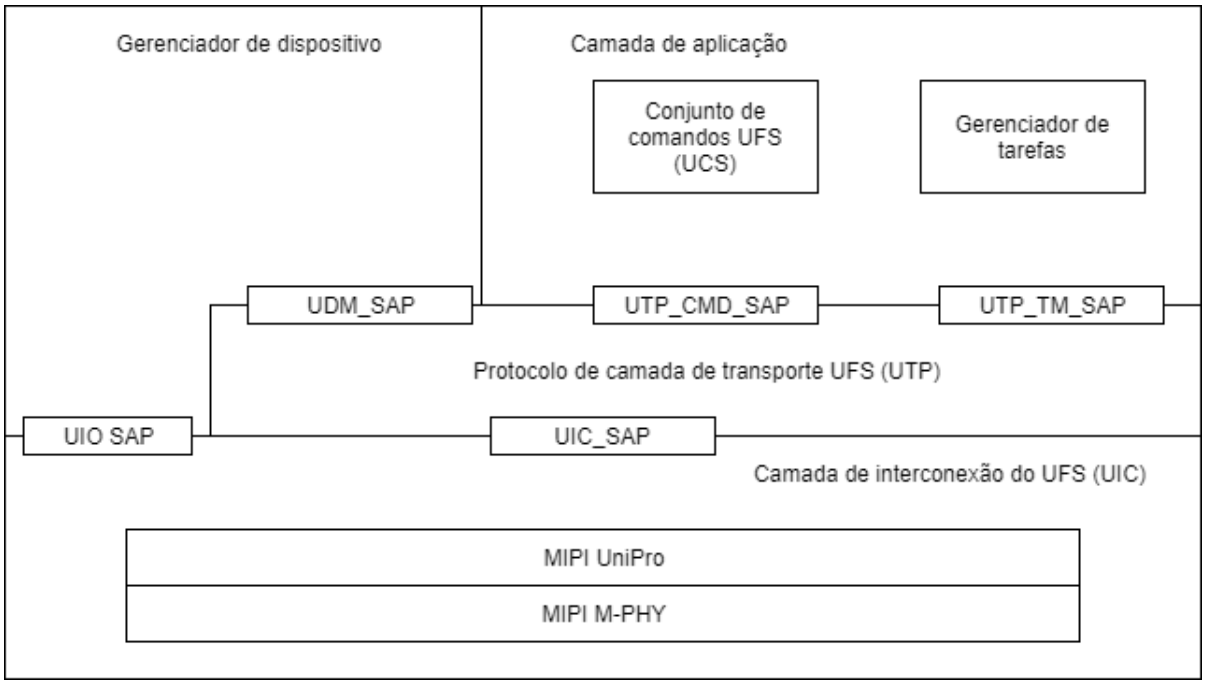


**Figura 2.5** Arquitetura de um sistema em alto nível que utiliza UFS

encapsular o protocolo em uma estrutura de moldura para ser transmitida pela camada de interconexão. A camada de interconexão tem IP digital e analógico responsáveis pela transmissão dos dados.

#### 2.4.2.1 Camada de aplicação

A camada de aplicação tem 3 principais componentes: A camada UCS, gerenciador de tarefas e gerenciador de dispositivos. A interface UFS utiliza como parâmetro o protocolo SCSI. UFS suporta um subconjunto de comandos SCSI definidos por SPC-4 e SBC-3. Os principais comandos são os seguintes:

- UCS lida com os comandos SCSI suportados pela especificação UFS.
- O gerenciador de tarefas que lida com funções de gerenciamento definidas pela especificação do UFS para o controle da fila de comando.
- O gerenciador de dispositivo lida com operações a nível de operações de dispositivo e operações de configuração do dispositivo. Operações de nível de dispositivo consiste nas operações de gerenciamento de energia e comandos para a camada de interconexão. Configurações a nível de dispositivo consiste gerenciar cada solicitação que são utilizadas para modificar e obter informação sobre a configuração do dispositivo.

#### 2.4.2.2 Camada de transporte

A camada de transporte é responsável de prover serviços para as camadas superiores via pontos de acesso de serviços (SAP), como mostrado na figura 2.5 UTP define 3 SAPs para camadas superiores:

- UDM_SAP: O SAP do gerenciador de dispositivo é para prover serviços para operações a nível de dispositivo. Estas operações são feitas por requisições.
- UTP_CMD_SAP: O comando SAP é responsável por prover serviços para a camada UCS de transportar comandos
- UTP_TM_SAP: o SAP de gerenciamento de tarefas é responsável por prover serviços para o gerenciador de tarefas para transportar as funções do gerenciador de tarefas. UTP transporta mensagens via protocolo de informação UFS, também conhecido como UPIU.

#### 2.4.2.3 Camada de interconexão

A camada de interconexão é a camada mais baixa da arquitetura de camadas do UFS. A camada é responsável pela comunicação entre o dispositivo UFS e o hospedeiro UFS. A UIC utiliza os protocolos M-PHY e UniPro e provê 2 SAPs para camadas superiores.

- UIC_SAP: UIC_SAP é responsável por prover serviços para transportar UPIU entre o hospedeiro UFS e o dispositivo UFS.
- UIO_SAP: UIO_SAP é responsável por prover serviços para delegar comandos para as camadas do UniPro.

### 2.4.3 Caracterísitcas da UFS

UFS consiste de oito unidades lógicas configuráveis, e 4 bem-conhecidas unidades lógicas. Uma unidade lógica é uma entidade endereçável externamente independente que processa os comandos e realiza funções de gerenciamento de tarefas. Cada unidade lógica pode ser configurada como unidade lógica de inicialização com no máximo duas. As unidades lógicas bem-conhecidas são: Inicialização, que é a representação virtual para a unidade lógica contendo o código de inicialização real, Relatório de unidades lógicas, Dispositivo UFS que disponibiliza o nível de interação com o dispositivo UFS (i.e. controle de gerenciamento de energia e RPMB provê função RPMB com seus próprios processos e espaço de memória.

O particionamento da memória UFS ocorre com a designação das unidades lógicas. O tipo de memória padrão é para características normais de memória, tipo de código de sistema para uma unidade lógica que não é atualizada frequentemente (arquivos de sistema, executáveis, etc.), as do tipo não-persistente é usado para informação temporária e o tipo de memória melhorada é deixado em aberto para necessidades específicas ou especificação do próprio fabricante.

Dispositivos UFS suportam sete estados de funcionamento que são controlados pelos comandos START, STOP, UNIT e alguns atributos. UFS possui quatro modos básicos que são: Ativo, hibernado, ocioso e desligado. Também possui três estados transicionais para facilitar a mudança de um estado para outro. Cada estado tem seu próprio perfil e UFS pode suportar até 16 configurações ativas. O hospedeiro pode escolher entre perfis pré-determinados ou até o próprio definido pelo usuário para entregar a maior performance.

UFS tem uma vantagem muito grande entre em relação as outras tecnologias aqui apresentadas que é a possibilidade de escrever e ler ao mesmo tempo, enquanto que nas outras memórias não é possível. Também há a fila de comando que é controlada pela UCS que organiza os comandos de forma que os comandos da memória sejam executados mais rápido.

A tabela 2.4 mostra uma comparação de forma mais clara entre as diferentes tecnologias apresentadas neste trabalho. Na próxima sessão, será explicado como foi realizado os experimentos.

**Tabela 2.3** Comparação entre as diferentes tecnologias de armazenamento flash

| | MMC | SD | eMMC | UFS |
|---|---|---|---|---|
| Org. da memória | 1) Página de escrita<br>2) Bloco de remoção<br>3) Bloco protegido de escrita | 1) Página de escrita<br>2) Bloco de remoção<br>3) Bloco protegido de escrita | 1) Página de escrita<br>2) Bloco de remoção<br>3) Bloco protegido de escrita | 1) 8 LUs independentes configuráveis<br>2) 4 LUs bem-conhecidas |
| Tecnologia | 2D | 2D | 2D | 2D |
| Cél. de memória | Flash (SLC) | Flash (SLC/MLC/TLC) | Flash (SLC) | Flash (SLC) |
| Part. de memória | Não é especificada pelo padrão | Não é especificada pelo padrão | 1) Área de inicialização<br>2) Uma partição RPMB (128kb)<br>3) 4 Partições de uso geral e áreas de aprimoramento de dados do usuário | 1) Partição de dados de múltiplos usuários<br>2) Partições de inicialização<br>3)Partição RPMB |
| Modos de operação | 1) Modo de identificação<br>2) Modo de interrupção<br>3) Modo de transmissão de dados | 1) Modo ocioso SD<br>2) Modo ocioso SPI<br>3) Modo de transmissão de dados | 1) Modo de inicialização<br>2) Modo de identificação<br>3) Modo de interrupção<br>4) Modo de transmissão de dados | 1) Ativo<br>2) Ocioso<br>3) Hibernado<br>4) Desligado<br>5) Pré-ativo<br>6) Pré-hibernação<br>7)Pré-desligamento |
| Números de pinos | 13 | 8 | 13 | 14 |
| Transmissão | Apenas síncrono | Apenas síncrono | Apenas síncrono | Síncrono Assíncrono |
| Tam. do comando | 48 bits | 48 bits | 48 bits | 128 bits |
| Qtd. de comandos | 64 (26 reservados) | 29 | 64 (21 reservados) | 27 |
| Respostas | Respostas diferentes para cada comando de 48 bits até 136 bits | Respostas diferentes para cada comando de 48 bits até 136 bits | 5 Respostas diferentes de um comando para outro | Resposta via UPIU (23bytes) |
| Interface | 1) CLK<br>2) RESET<br>3) linha CMD 1-bit (bidirecionall)<br>4) linha de dados 8-bits | 1) CLK<br>2) RESET<br>3) linha CMD 1-bit (bidirecionall)<br>4) linha de dados 8-bits | 1) CLK<br>2) Reset<br>3) linha CMD 1-bit (bidirecional)<br>4)linha de dados 8-bits | 1) CLK<br>2) Reset<br>3) Faixa para upstream e outra para downstream<br>4) I/O diferencial verdadeiro e sinal par complementar |
| Tipo de interface | Paralela | Paralela | Paralela | Serial |
| Freq. de operação | 52MHz | 208MHz | 200MHz | 19.2/26/38.4/52 MHz (dependendo do modo de operação) |
| Ger. de energia | Nenhum | Modo hibernado | Apenas modo hibernado | 1) Modo hibernado<br>2) Modo desligado |

CAPÍTULO 3

# Resultados Experimentais

Neste capítulo será detalhado como foi pensado e implementado os experimentos realizados a fim de observar como as tecnologias de memória flash se comportam em celulares para operações de escrita, leitura e remoção de dados.

## 3.1 Projeto de experimentos

Os experimentos foram realizados utilizando-se de duas ferramentas desenvolvidas e celulares *Android* para teste.

### 3.1.1 Ferramentas

Foi desenvolvido um programa escrito em C que possui 3 funções e recebe parâmetros dependendo da função escolhida. O programa será chamado a partir de agora de *stresser*. O *stresser* recebe parâmetros para saber qual comando executar. Os comandos podem ser

- write <tamanho> <padrão> : O programa escreve na memória do celular um arquivo chamado stresser.str que é uma string de 1s e 0s de tamanho especificado pelo usuário em kb e o programa possui dois padrões de substring para escrever e retorna o tempo estimado de escrita deste arquivo em microssegundos.

- read: O programa retorna o tempo de leitura em microssegundos do arquivo stresser.str já escrito na memória. Retorna um erro se não existir o arquivo stresser.str.

- delete: O programa retorna o tempo de remoção do arquivo stresser.str já escrito na memória. Retorna um erro se não existir o arquivo stresser.str.

Este programa em C é utilizado por um programa maior escrito em Java, que identifica celulares conectados ao computador via uma série de comandos adb. Uma vez selecionado o celular (ou vários) é realizado uma série de comandos, tais como colocar o celular em modo root, instalar o *stresser* no celular para poder ser executável através de comandos adb, autorizar a pasta no sistema de arquivos para o *stresser* poder escrever a string, para preparar o celular para uma série de ciclos de escrita, leitura e remoção.

Este programa então começa a estressar o celular de acordo com uma série de parâmetros, como o tamanho do arquivo a ser avaliado a cada ciclo ou se será pelo tamanho do bloco que é o padrão no sistema de memória. O teste todo pode ser configurado entre a quantidade de ciclos,

quantidade de bad blocks detectados, porcentagem de bad blocks identificados em relação a quantidade de blocos total na memória ou até falhar (totalidade de bad blocks).

A ferramenta também permite coletar informações sobre a memória do celular selecionado, stress paralelo (vários celulares executando ciclos em paralelo) e até desativar a proteção térmica do próprio celular para avaliar temperatura do hardware quando há uma grande demanda de leitura/escrita. Estes parâmetros são encontrados olhando arquivos disponíveis na partição /dev do *Android*. Além do comando *mount* do adb, em que é mostrado o tamanho de cada partição (estamos interessados na partição /data, pois é lá onde o *stresser* irá escrever).

Quando a ferramenta inicia, primeiramente é coletada uma série de informações sobre a memória e o sistema que está sendo avaliado. Como informações sobre a versão do *Android*, número de blocos, páginas, número de bad blocks, capacidade da memória, tamanho do bloco de remoção etc. Em seguida, é inciado um *ciclo*. O *ciclo* nada mais é do que a escrita do arquivo na memória pelo *stresser* e então é guardado o tempo de escrita, depois este arquivo é lida pelo *stresser* e é guardado o tempo de leitura. Finalmente, este arquivo é deletado e o *stresser* coleta o tempo que levou para o arquivo ser removido.

Ao fim de cada ciclo é guardado uma série de métricas que depois são exportadas para um arquivo do tipo CSV e ao fim do conjunto de ciclos (por alguma das condições de parada) este arquivo é gravado no computador e pode ser usado para futura análise com ferramentas como o Minitab ou Excel. Estas métricas retiradas de cada bateria de ciclos, incluem os tempos de escrita, leitura e remoção discutidas neste trabalho. O CSV é composto de informações sobre o dispositivo e também informações sobre cada ciclo. informações de cada ciclo é representado por uma linha do CSV.

A ferramenta também possui interface gráfica com gráficos criados em tempo real para uma representação visual para checar o progresso durante a execução dos ciclos.

### 3.1.2 Dispositivos a serem testados

Os dispositivos a serem testados foram dois celulares diferentes, ambos executando *Android* 7.1.1. O sistema de arquivos na partição /data nos dispositivos é o chamado F2FS (Flash Friendly File System) desenvolvido pela Samsung Electronics e Motorola Mobility que foi criado pensando desde o começo para a aplicação em sistemas de memória flash NAND o que possui algumas otimizações pensando em problemas inerentes a sistemas de memória flash como separação de dados quente (altamente utilizado) e frio (com pouco acesso) e também implementa dois tipos diferentes de garbage collection.

O celular número 1 tem como sistema operacional o *Android* 7.1.1 com um sistema de memória eMMC com 16GB de armazenamento de um chip fabricado pela Samsung Electronics. O celular 1 possui o processador 1.2 GHz Qualcomm® Snapdragon™ 410 quad-core com 450 MHz Adreno 306 GPU com 2GB RAM.

O celular 2 é tem como sistema operacional o *Android* 7.1.1 com um sistema de memória UFS com 32GB de armazenamento de um chip fabricado pela Samsung Electronics. O celular 2 possui um processador Qualcomm® Snapdragon™ 820 com até 1.8 GHz Quad-core CPU e Adreno 530 GPU e um processador de computação de linguagem natrual contextual.

Como dito, o fato de usarem o mesmo sistema operacional e o mesmo sistema de arquivos tem como objetivo eliminar variáveis para focar apenas em qual sistema de memória flash estão

usando.

Ambos também foram equipados com um chip de memória microSD da SanDisk com capacidade de 2GB para manter a isonomia entre diferentes fabricantes de microSD.

### 3.1.3 Descrição dos experimentos

Os experimentos realizados foram os seguintes: Primeiro, foi separado entre a tecnologia de armazenamento empregada. Ou seja, foram executados experimentos para a memória eMMC, UFS e SD. No caso do SD, foi realizado em ambos os celulares a fim de mitigar capacidades de processamento do celular e apenas focar no sistema de memória SD. Para cada sistema de memória, foram realizados dois experimentos com dois tamanhos diferentes de arquivo. Um com 4KB e outro com 1024KB. Na realização destes experimentos, foi utilizado uma rodada de 30 ciclos de escrita/leitura/remoção para cada a fim de obter resultados mais precisos para então fazer uma média dessas 30 iterações. Finalmente, foram obtidos os resultados de tempo de leitura, escrita e remoção médio em cada uma destas situações.

## 3.2 Resultados dos experimentos

Como dito, os experimentos a seguir serão usados para avaliar e comparar as diferentes tecnologias de armazenamento utilizada em dispositivos móveis.

### 3.2.1 Experimento 1: Tempo de leitura 4KB

No experimento 1, O tempo tempo médio de leitura na memória interna, que utiliza eMMC foi de 42,933 us com variância de 39,7574 até 46,1093 . O celular 2, o tempo médio de leitura na memória interna que utiliza UFS 1.0 foi de 66,5172 us com variância de 57,5794 a 75,4551. O tempo médio de leitura na memória externa no celular 1, utilizando SD Card foi de 56,3667 us, com variância de 37,5866 a 75,1467. E o tempo médio de leitura na memória externa no celular 2, utilizando SD Card foi de 65,633 us, com variância de 59,4793 a 71,7874. É possível notar que o celular 1, com especificações inferiores, teve um desempenho melhor tanto na leitura de memória interna (eMMC) quanto na memória externa (SD card). Os resultado do experimento 1 pode ser visto na figura 3.1.

Depois de exibido de forma gráfica, foi realizado um teste ANOVA (Análise de variância) com nível de significância 0,05, para identificar se os resultados eram estatisticamente diferentes ou não. De acordo com o resultado obtido após a realização da ANOVA, podemos verificar que:

- No celular 1, o experimento realizado na memória interna e na memória externa são estatisticamente iguais.

- O celular 1, executando o experimento 1 na sua memória interna (eMMC) é estatisticamente diferente ao celular 2 executando o experimento 1 na sua memória interna (UFS). Com resultados melhores obtidos no celular 1 com eMMC.

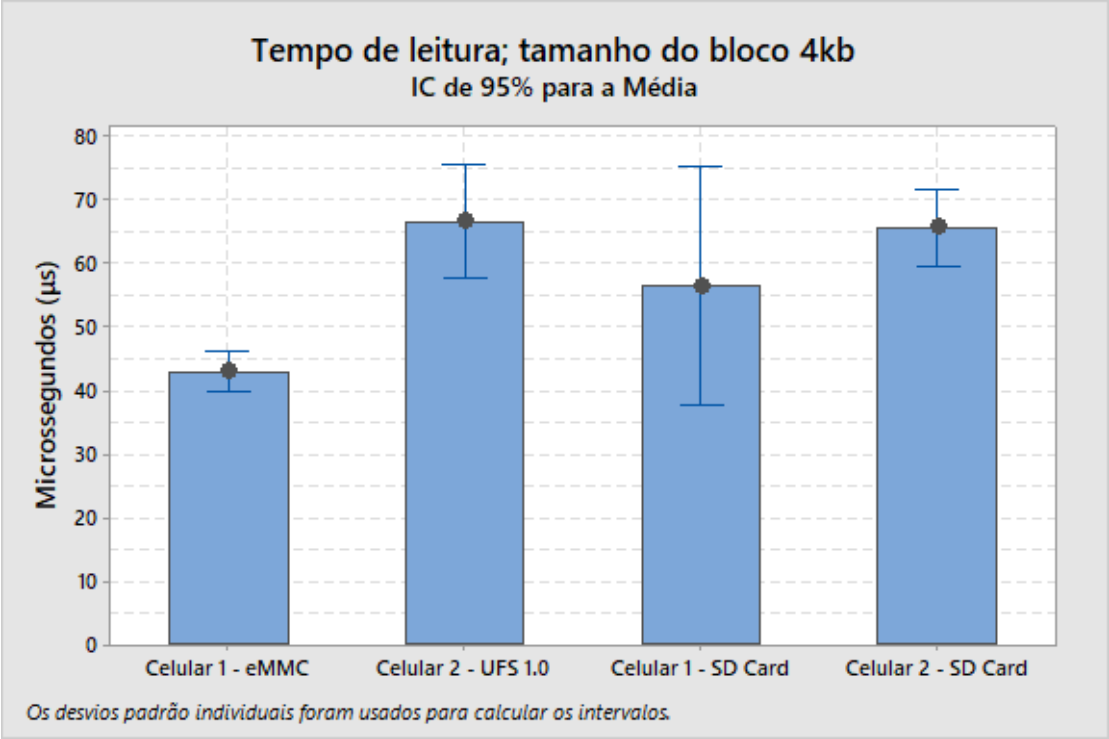


**Figura 3.1** Tempo de leitura, tamanho do arquivo 4kb

- No celular 2, o experimento realizado na memória interna e na memória externa são estatisticamente iguais.
- O resultado do experimento do celular 2 é estatisticamente igual ao resultado do experimento no celular 1 utilizando a memória externa.

### 3.2.2 Experimento 2: Tempo de escrita 4KB

No experimento 2, O tempo tempo médio de escrita na memória interna, que utiliza eMMC foi de 118,733 us com variância de 95,6337 até 141,833. O celular 2, o tempo médio de escrita na memória interna que utiliza UFS 1.0 foi de 184,5 us com variância de 139,065 a 229,935. O tempo médio de escrita na memória externa no celular 1, utilizando SD Card foi de 104,9 us, com variância de 87,9645 a 121,835. E o tempo médio de escrita na memória externa no celular 2, utilizando SD Card foi de 183,333 us, com variância de 139,409 a 227,257. É possível notar que o celular 1, com especificações inferiores, teve um desempenho melhor tanto na leitura de memória interna (eMMC) quanto na memória externa (SD card). Naturalmente, também foi realizado um teste ANOVA com nível de significância 0,05, para identificar se os resultados eram estatisticamente diferentes ou não. De acordo com o resultado obtido após a realização da ANOVA, podemos verificar que:

- No celular 1, o experimento realizado na memória interna e na memória externa são estatisticamente iguais.

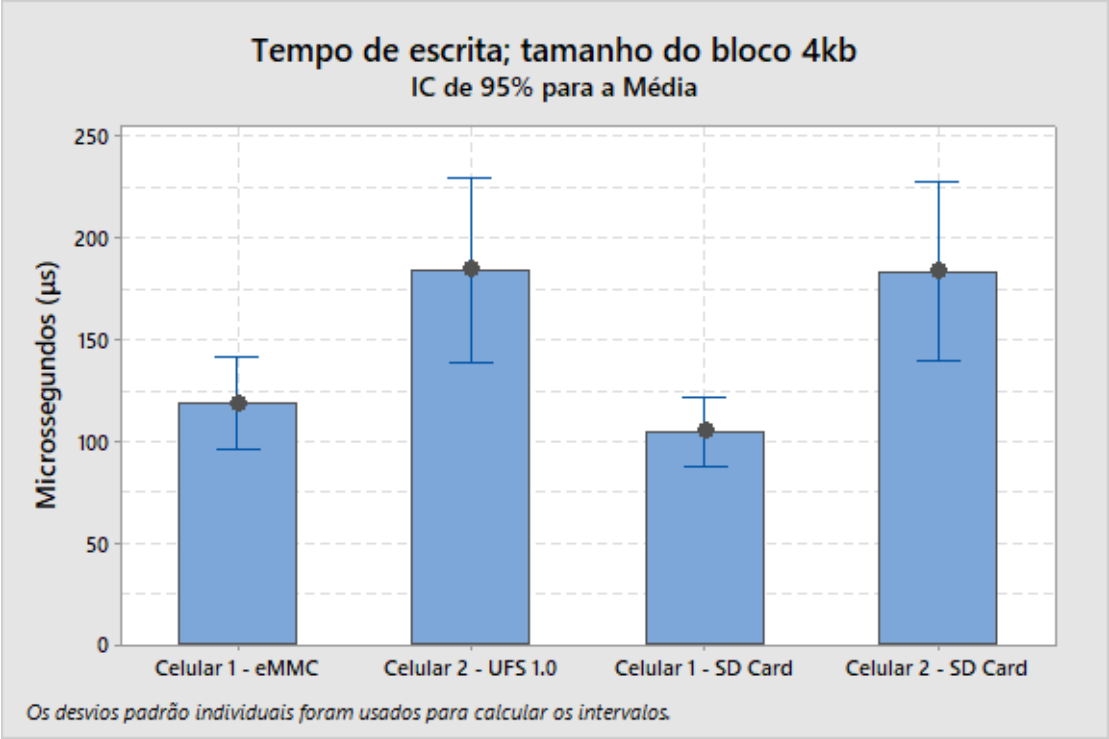


**Figura 3.2** Tempo de escrita, tamanho do arquivo 4kb

- O celular 1, executando o experimento 2 na sua memória interna (eMMC) é estatisticamente diferente ao celular 2 executando o experimento 1 na sua memória interna (UFS). Com resultados melhores obtidos no celular 1 com eMMC.
- No celular 2, o experimento realizado na memória interna e na memória externa são estatisticamente iguais.
- O celular 1, executando o experimento 2 na sua memória externa é estatisticamente diferente ao celular 2 executando o experimento 2 na sua memória externa. Com resultados melhores obtidos no celular 1.

### 3.2.3 Experimento 3: Tempo de remoção 4KB

No experimento 3, O tempo tempo médio de remoção na memória interna, que utiliza eMMC foi de 116 us com variância de 98,4115 até 133,589. O celular 2, o tempo médio de remoção na memória interna que utiliza UFS 1.0 foi de 251,621 us com variância de 226,129 a 277,112. O tempo médio de remoção na memória externa no celular 1, utilizando SD Card foi de 118,3 us, com variância de 105,514 a 131,086. E o tempo médio de escrita na memória externa no celular 2, utilizando SD Card foi de 251,1 us, com variância de 231,478 a 270,722. É possível notar que o celular 1, com especificações inferiores, teve um desempenho melhor tanto na leitura de memória interna (eMMC) quanto na memória externa (SD card).

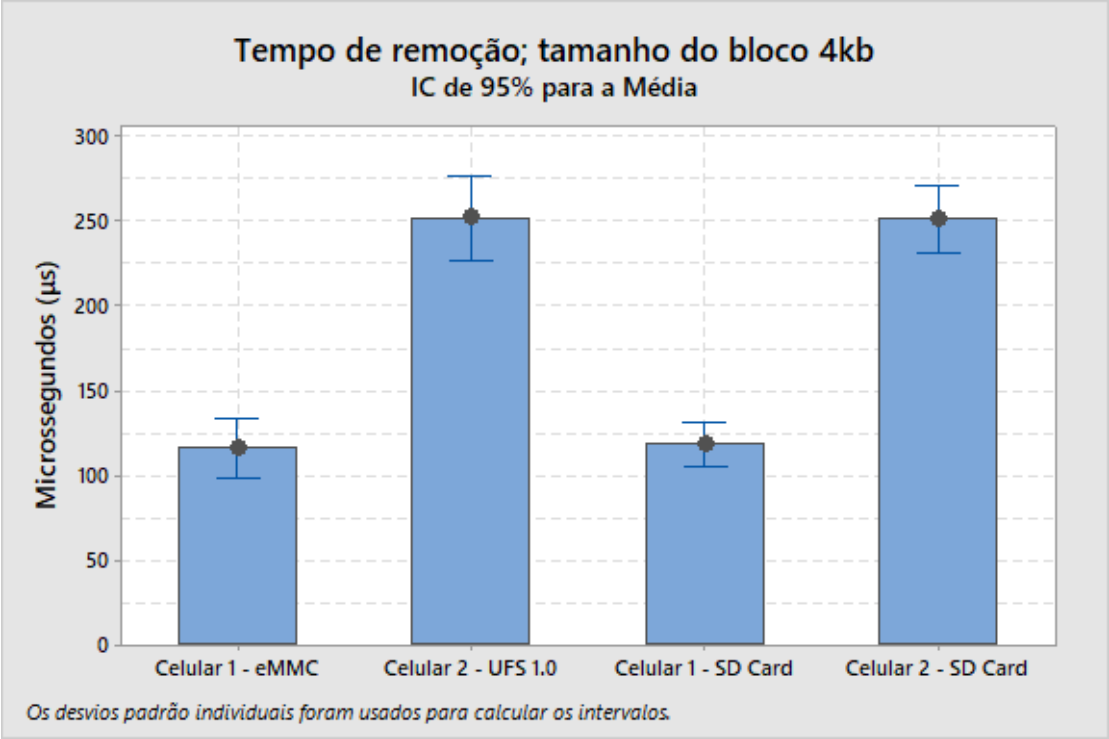


**Figura 3.3** Tempo de remoção, tamanho do arquivo 4kb

Mais uma vez foi realizado um teste ANOVA com nível de significância 0,05 (Análise de variância) para identificar se os resultados eram estatisticamente diferentes ou não. De acordo com o resultado obtido após a realização da ANOVA, podemos verificar que:

- No celular 1, o experimento realizado na memória interna e na memória externa são estatisticamente iguais.
- O celular 1, executando o experimento 2 na sua memória interna (eMMC) é estatisticamente diferente ao celular 2 executando o experimento 1 na sua memória interna (UFS). Com resultados melhores obtidos no celular 1 com eMMC.
- No celular 2, o experimento realizado na memória interna e na memória externa são estatisticamente iguais.
- O celular 1, executando o experimento 3 na sua memória externa é estatisticamente diferente ao celular 2 executando o experimento 3 na sua memória externa. Com resultados melhores obtidos no celular 1.

### 3.2.4 Experimento 4: Tempo de leitura 1024KB

No experimento 4, O tempo tempo médio de leitura na memória interna, que utiliza eMMC foi de 3496,27 us com variância de 2952,82 até 4039,71. O celular 2, o tempo médio de leitura na

memória interna que utiliza UFS 1.0 foi de 2890,7 us com variância de 2539,86 a 3241,54. O tempo médio de leitura na memória externa no celular 1, utilizando SD Card foi de 3359,63 us, com variância de 2918,82 a 3800,45. E o tempo médio de leitura na memória externa no celular 2, utilizando SD Card foi de 3022,07 us, com variância de 2853,39 a 3190,75. É possível notar que o celular teve um desempenho melhor tanto na leitura de memória interna (UFS) quanto na memória externa (SD card).

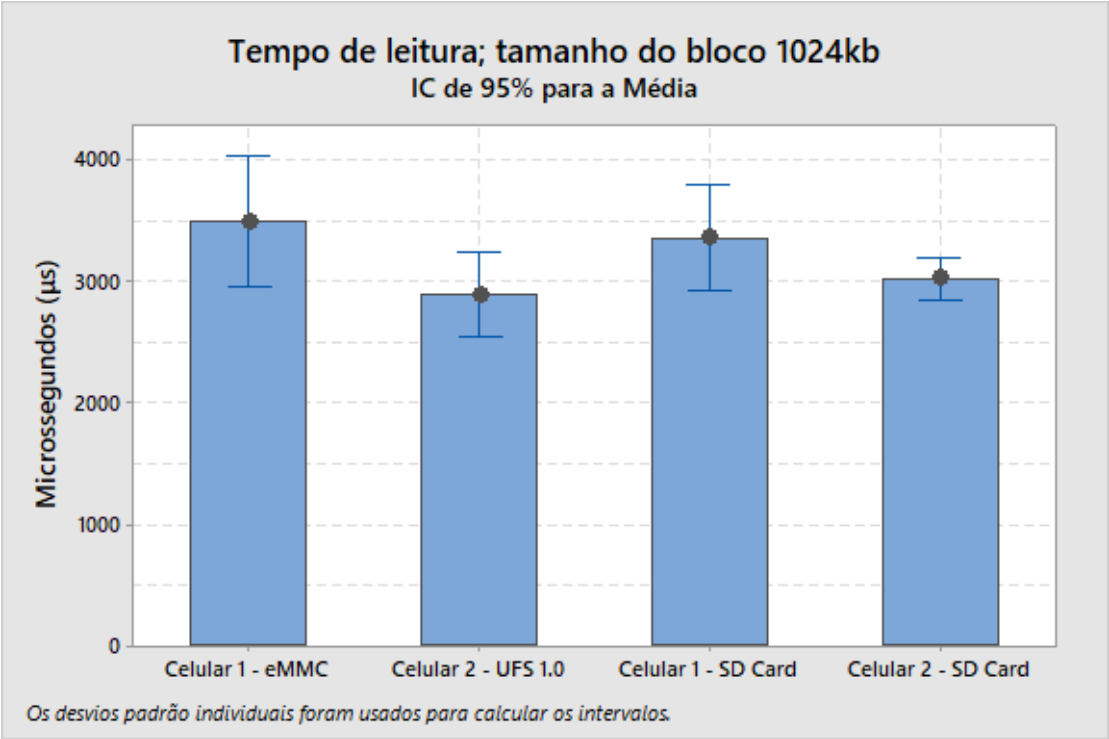


**Figura 3.4** Tempo de leitura, tamanho do arquivo 1024kb

Foi realizado um teste ANOVA com nível de significância 0,05, para identificar se os resultados eram estatisticamente diferentes ou não. De acordo com o resultado obtido após a realização da ANOVA com nível de significância 0,05, podemos verificar que como os resultados muito parecidos e com todas as variâncias na mesma interseção. Não podemos afirmar que os resultados sejam diferentes a fim de apontar quem teve melhor desempenho. Assim é necessário afirmar que no experimento 4 eles foram estatisticamente iguais.

### 3.2.5 Experimento 5: Tempo de escrita 1024KB

No experimento 5, O tempo tempo médio de escrita na memória interna, que utiliza eMMC foi de 3812,23 us com variância de 3689,05 até 3935,42. O celular 2, o tempo médio de escrita na memória interna que utiliza UFS 1.0 foi de 3415,2 us com variância de 3269,33 a 3561,07. O tempo médio de escrita na memória externa no celular 1, utilizando SD Card foi de 3938,83 us, com variância de 3767,46 a 4110,20. E o tempo médio de escrita na memória externa no celular

2, utilizando SD Card foi de 3615,6 us, com variância de 3475,39 a 3755,81. É possível notar que o celular teve um desempenho melhor tanto na leitura de memória interna (UFS) quanto na memória externa (SD card).

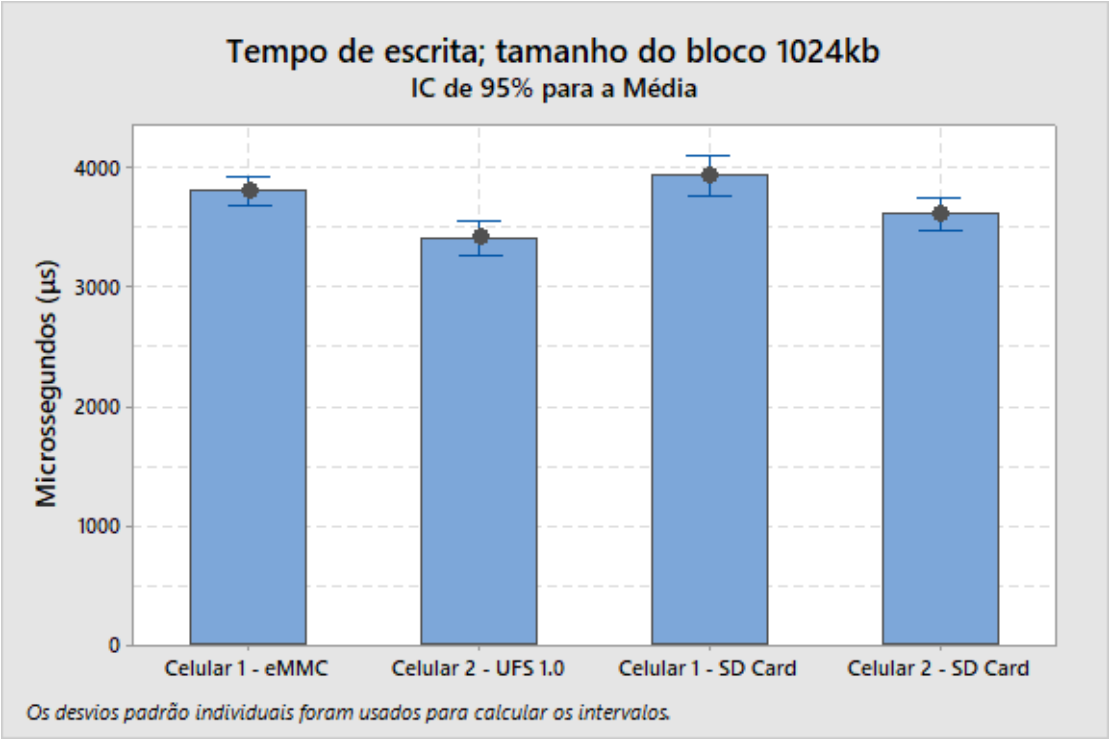


**Figura 3.5** Tempo de escrita, tamanho do arquivo 1024kb

Mais uma vez foi realizado um teste ANOVA com nível de significância 0,05, para identificar se os resultados eram estatisticamente diferentes ou não. De acordo com o resultado obtido após a realização da ANOVA, podemos verificar que:

- No celular 1, o experimento realizado na memória interna e na memória externa são estatisticamente iguais.
- O celular 1, executando o experimento 5 na sua memória interna (eMMC) é estatisticamente diferente ao celular 2 executando o experimento 5 na sua memória interna (UFS). Com resultados melhores obtidos no celular 2.
- No celular 2, o experimento realizado na memória interna e na memória externa são estatisticamente iguais.
- O celular 1, executando o experimento 5 na sua memória externa é estatisticamente diferente ao celular 2 executando o experimento 5 na sua memória externa. Com resultados melhores obtidos no celular 2.
- O celular 1, executando o experimento 5 na sua memória interna é estatisticamente igual ao celular 2 executando o experimento 5 na sua memória externa.

### 3.2.6 Experimento 6: Tempo de remoção 1024KB

No experimento 6, O tempo tempo médio de remoção na memória interna, que utiliza eMMC foi de 1115,03 us com variância de 934,997 até 1295,07. O celular 2, o tempo médio de remoção na memória interna que utiliza UFS 1.0 foi de 1428,63 us com variância de 1194,64 a 1662,63. O tempo médio de remoção na memória externa no celular 1, utilizando SD Card foi de 1042,93 us, com variância de 877,998 a 1207,88. E o tempo médio de escrita na memória externa no celular 2, utilizando SD Card foi de 1557,57 us, com variância de 1343,08 a 1772,05 . É possível notar que o celular 1, com especificações inferiores, teve um desempenho melhor tanto na leitura de memória interna (eMMC) quanto na memória externa (SD card).

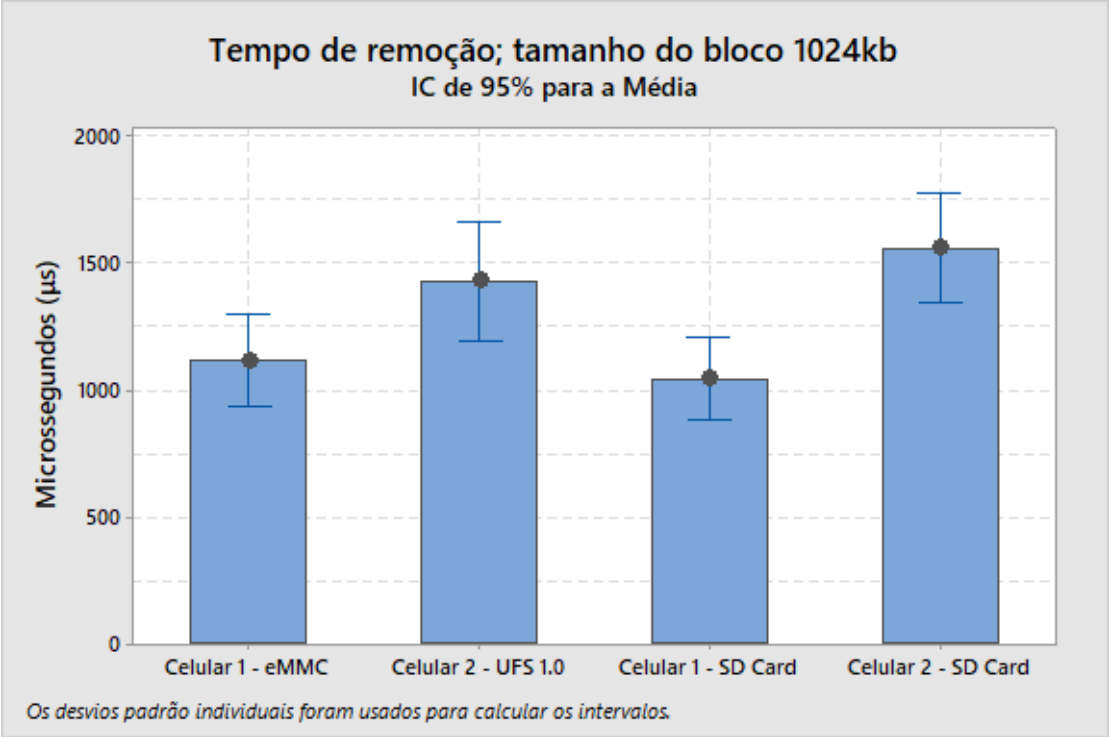


**Figura 3.6** Tempo de remoção, tamanho do arquivo 1024kb

Novamente foi realizado um teste ANOVA com nível de significância 0,05, para identificar se os resultados eram estatisticamente diferentes ou não. De acordo com o resultado obtido após a realização da ANOVA, podemos verificar que:

- No celular 1, o experimento realizado na memória interna e na memória externa são estatisticamente iguais.
- O celular 1, executando o experimento 6 na sua memória interna (eMMC) é estatisticamente igual ao celular 2 executando o experimento 6 na sua memória interna (UFS).
- No celular 2, o experimento realizado na memória interna e na memória externa são estatisticamente iguais.

- O celular 1, executando o experimento 6 na sua memória externa é estatisticamente diferente ao celular 2 executando o experimento 6 na sua memória externa. Com resultados melhores obtidos no celular 1.

## 3.3 Interpretações dos resultados

Os resultados obtidos nos experimentos mostram algumas coisas bastante interessantes. Primeiro, os resultados para um mesmo celular na memória externa e interna em um dado experimento sempre foram estatisticamente iguais. Apesar de ter uma leve piora em números absolutos, não é o suficiente para afirmar alguma diferença. Quando o tamanho do arquivo é de 4kb, existe uma tendência de que a eMMC se comporte melhor do que a UFS. Isto pode ser explicado em parte pela diferença do tamanho do bloco entre a memória eMMC e a memória UFS e seus controladores. O tamanho do bloco na memória eMMC é de apenas 512 bytes, enquanto que na memória UFS é 4096 bytes, então quando se procura por um arquivo na memória eMMC é feito uma operação sequencial, mais rápida que acesso randômico (na UFS) pois cabe em um único bloco. Isto fica bem mais explicito nas operações de remoção pois para deletar um arquivo é necessário remover todo um bloco e um bloco de tamanho maior, de 4096 bytes é muito mais custoso de se remover do que um de 512 bytes. Assim, nos experimentos de 4kb, a eMMC possui um desempenho melhor do que o da UFS em leitura, porém o desempenho da UFS é estatisticamente igual ao da memória externa no celular 1. Na escrita, o comportamento foi repetido e o celular 1 obteve um desempenho melhor que o celular 2 tanto na memória interna quanto na memória externa.

Nos experimentos com tamanho de 1024kb, houve uma mudança de comportamento. A tendência se inverteu e na maioria dos casos, o celular 2 com UFS obteve uma performance melhor. No caso da leitura do arquivo de 1024kb, não foi possível afirmar que nenhum das tecnologias e celulares eram diferentes entre si. Porém na escrita é possível afirmar que celular 2 obteve um resultado melhor comparando as memórias internas.

Pelo mesmo motivo descrito acima no caso dos experimentos com tamanho do bloco da memória mostra que a UFS tem uma série de otimizações para lidar com arquivos de maior tamanho. Faz sentido então celulares mais novos e com maior capacidade estarem utilizando UFS em vez de eMMC pois com o maior tráfego de dados e de consumo de mídia em dispositivos móveis, é possível afirmar com os experimentos descritos que o UFS proporciona uma experiência de usuário melhor para usuários de celulares High-end. E possivelmente essa tecnologia será difundida entre celulares de menor custo no futuro.

CAPÍTULO 4

# Conclusões

Este trabalho teve como objetivo uma avaliação das diferentes tecnologias de memória flash não-volátil em celulares. Os experimentos foram realizados em dois celulares *Android* com duas tecnologias de memória não-volátil internas diferentes além de uma avaliação de um cartão de memória externa.

Os sistemas foram avaliados por uma ferramenta desenvolvida em parceria com o convênio CIn/Motorola e então os resultados dos experimentos foram apresentados e devidamente interpretados.

O que foi importante de avaliar é que eMMC, com suas constantes evoluções ainda é um sistema bastante robusto e com um bom desempenho. É compreensível porque ainda é o sistema mais utilizado no mercado. Importante também ver como UFS tem um melhor desempenho conforme aumenta a quantidade de dados. Mostrando que o mercado aposta em uma maior quantidade de dados trafegando através dos dispositivos móveis. Com o seu baixo consumo e velocidade maiores, com certeza será um avanço na capacidade de tarefas que um celular pode fazer além de uma melhor experiência de usuário com repostas mais rápidas e uma maior autonomia de bateria.

## 4.1 Trabalhos futuros

Como trabalhos futuros, seria interessante realizar experimentos com padrões mais novos das respectivas tecnologias, como UFS 3.0 e eMMC 5.1, além de versões do sistema *Android* mais novas para identificar melhoras em relação ao mesmo hardware ou não.

A Samsung Electronics lançou recentemente cartões de memória UFS em detrimento ao SD Card. Também seria de grande valia a diferença entre UFS como memória interna e UFS como memória externa se há alguma diferença de desempenho e até entre o SD card e o UFS externo no mesmo hardware.

Outro ponto de melhoria seria uma quantidade maior de experimentos e duplicidade de hardware para garantir que não houve hardwares defeituosos ou sistema operacional com algum erro.

# Referências Bibliográficas


[ACCS15] Paolo Amato, Danilo Caraccio, Emanuele Confalonieri, and Marco Sforzin. An analytical model of emmc key performance indicators. In *Memory Workshop (IMW), 2015 IEEE International*, pages 1–4. IEEE, 2015.

[AP+11] SD Card Association, SD Specifications Part, et al. A2: Sd host controller simplified specification version 2.00, 2011.

[BG11] Joe Brewer and Manzur Gill. *Nonvolatile memory technologies with emphasis on flash: a comprehensive guide to understanding and using flash memory devices*, volume 8. John Wiley & Sons, 2011.

[CAB+15] E Confalonieri, P Amato, D Balluchi, D Caraccio, and M Dallabora. Mobile memory systems. In *Mobile Systems Technologies Workshop (MST), 2015*, pages 1–7. IEEE, 2015.

[eMM] emmc to ufs: How nand memory for mobile products is evolving – samsung global newsroom. `https://news.samsung.com/global/emmc-to-ufs-how-nand-memory-for-mobile-products-is-evolving`. (Accessed on 11/26/2018).

[GA+] SD Group, SD Card Association, et al. Sd specifications part 1 physical layer specification.

[id:07] *[IEEE Press Series on Microelectronic Systems] Nonvolatile Memory Technologies with Emphasis on Flash || Introduction to Nonvolatile Memory*, volume 10.1002/9780470181355. 2007.

[Mis] Vijayakrishnan; Singh Neeraj Kumar Mishra, Sanjeeb; Rousseau. *System on Chip Interfaces for Low Power Design*.

[Mob] • mobile os market share 2018 | statista. `https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/`. (Accessed on 11/27/2018).

[Sam] Samsung dex | apps | samsung br. `https://www.samsung.com/br/apps/samsung-dex/`. (Accessed on 11/27/2018).

[Sta11] JEDEC Standard. Universal flash storage (ufs). *JESD220, JEDEC Solid State Technology Association*, 2011.

[Sta15] JEDEC Standard. Embedded multi-media card(emmc) electrical standard (5.1). *JESD84-B51, JEDEC Solid State Technology Association*, 2015.

Este volume foi tipografado em LaTeX na classe UFPEThesis (www.cin.ufpe.br/~paguso/ufpethesis).