PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DE COMPUTAÇÃO

Eliminação Segura de Arquivos em Memória Não-Volátil

JULIA SILVA WEBER

Dissertação submetida para a Pontifícia Universidade Católica do Rio Grande do Sul como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Orientador: Prof. Dr. Avelino Zorzo

Porto Alegre, Brasil
Janeiro 2017

Ficha Catalográfica

W374e Weber, Julia Silva

Eliminação Segura de Arquivos em Memória Não-Volátil / Julia Silva Weber . – 2017.

108 f.

Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Avelino Francisco Zorzo.

1. Memória flash. 2. Remoção segura de arquivos. 3. Sobrescrita e Apagamento. I. Zorzo, Avelino Francisco. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS com os dados fornecidos pelo(a) autor(a).

Julia Silva Weber

Eliminação Segura de Arquivos em Memória Não-Volátil

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação, Faculdade de Informática da Pontifícia Universidade Católica do Rio Grande do Sul.

Aprovado em 21 de março de 2017.

BANCA EXAMINADORA:

Prof. Dr. Luciano Paschoal Gaspary (PPGC/UFRGS)

Prof. Dr. César Augusto Missio Marcon (PPGCC/PUCRS)

Prof. Dr. Avelino Francisco Zorzo (PPGCC/PUCRS - Orientador)

ELIMINAÇÃO SEGURA DE ARQUIVOS EM MEMÓRIA NÃO-VOLÁTIL

RESUMO

O advento da Internet das Coisas (IoT) e a popularização de dispositivos móveis com memória não-volátil traz novos desafios quanto a remoção de arquivos. Técnicas tradicionalmente empregadas em meios magnéticos não são efetivas quando aplicadas para memórias não voláteis, como a memória *flash*. Devido às características peculiares deste tipo de memória, notadamente a existência de uma Camada de Tradução da *Flash* (FTL), sistemas operacionais somente gerenciam blocos lógicos, e não tem mais controle direto dos blocos físicos de uma memória *flash*. Consequentemente, novos métodos de remoção segura foram desenvolvidos, que empregam operações de Sobrescrita com Zeros, de Apagamento de Blocos e técnicas de Apagamento Criptográfico.

Este trabalho analisa estes métodos, compara suas operações e propõe um novo método, com melhor desempenho que os descritos na literatura. O método proposto é um método híbrido, que combina de forma equilibrada operações de sobrescrita e apagamento, para evitar o apagamento desnecessário de blocos ainda não utilizados e reduzir o desgaste prematuro da memória. Para verificar a eficiência do método proposto e dos demais métodos, foi desenvolvido um simulador para exercitar a remoção de arquivos em diversos experimentos.

Palavras Chave: Memória *flash*, Remoção segura de arquivos, Sobrescrita e Apagamento.

SECURE FILE DELETION IN NON VOLATILE MEMORY

ABSTRACT

The advent of the Internet of Things (IoT) and the popularization of mobile devices with non-volatile memory brings new challenges regarding the removal of files. Techniques traditionally employed in magnetic media are not effective when applied to non-volatile memories, such as flash memory. Because of the peculiar characteristics of this type of memory, notably the existence of a Flash Translation Layer (FTL), operating systems only manage logical blocks, and no longer have direct control of the physical blocks of a flash memory. Consequently, new methods of safe removal have been developed, which employ Zero Override, Block Erase, and Cryptographic Erase techniques.

This work analyzes these methods, compares their operations and proposes a new method, with better performance than those described in the literature. The proposed method is a hybrid method, which combines overwriting and deletion operations to obtain a balanced use of these operations, avoid unnecessary deletion of unused blocks and reduce premature memory wear. To verify the efficiency of the proposed method and of the other methods, a simulator was developed to exercise the removal of files in several experiments.

Keywords: Flash Memory, Secure File Removal, Overwriting and Erasing.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BAM	Block Allocation Map
CE	Cryptographic Erase
CIS	Card Information Structure
ECC	Error correcting Code
EDAC	Error Detection And Correction
EU	Erase Unit
EUH	Erase Unit Header
Ext3	Third Extended Filesystem
FAT32	File Allocation Table 32
FTL	Flash Translation Layer
IoT	Internet of Things
MTD	Memory Technology Device
NVM	Non Volatile Memory
NTFS	New Technology File System
PCMCIA	Personal Computer Memory Card International Association
RAM	Random Access Memory
SSD	Solid State Drive
USB	Universal Serial Bus
VBM	Virtual Block Map
VPM	Virtual Page Map
ZO-A	Método de Sobrescrita (ZO) e Apagamento (A)
ZO-AB	Método de Sobrescrita (ZO) e Apagamento com Benefício (AB)
ZO-ABP	Método de Sobrescrita (ZO) e Apagamento com Benefício e Penalidade (ABP)
ZO-AP	Método de Sobrescrita (ZO) e Apagamento com Penalidade (AP)

ZOP-A Método de Sobrescrita com Penalidade (ZOP) e Apagamento (A)

ZOP-AB Método de Sobrescrita com Penalidade (ZO) e Apagamento com Benefício (AB)

ZOP-ABP Método de Sobrescrita com Penalidade (ZOP) e Apagamento com Benefício e Penalidade (ABP)

ZOP-AP Método de Sobrescrita com Penalidade (ZOP) e Apagamento com Penalidade (AP)

SUMÁRIO

1 INTRODUÇÃO

Memória não-volátil (Non Volatile Memory – NVM) é um tipo especial de memória que pode reter informações mesmo em caso de perda de energia [38]. NVM é tipicamente utilizada como armazenamento secundário, ou armazenamento persistente a longo prazo. Exemplos de NVM incluem memória de somente leitura, memória flash, dispositivos de armazenamento magnético (por exemplo, discos rígidos e fitas magnéticas) e discos ópticos. O principal uso da NVM é como um meio de armazenamento, e estudos recentes [38] [32] [42] sugerem a utilização de NVM implementada em dispositivos semicondutores para sistemas de arquivos ou até mesmo que sistemas de arquivos possam ser totalmente implementados em NVM. Entre estes tipos de memória, a memória flash se destaca, pois fornece grande capacidade de armazenamento de dados que é persistente e que pode ser acessado mais rapidamente do que os discos rígidos tradicionais. Entretanto, esse tipo de memória ainda é mais lenta do que a memória semicondutora tradicional (RAM).

A primeira memória *flash* comercial foi introduzida pela Intel Corporation em 1988, e era do tipo NOR. Memórias *flash* podem ser de dois tipos: NAND e NOR [53]. Memória tipo NOR tem tempos de escrita e apagamento relativamente longos, mas opera com endereços completos, permitindo assim o acesso aleatório a qualquer posição de memória. Em contrapartida, memória *flash* tipo NAND apresenta tempos menores para as operações de escrita e apagamento de dados e requer menos área do *chip* por célula, permitindo assim uma maior densidade de armazenamento e menor custo do que a do tipo NOR. Contudo o tipo NAND não permite o acesso aleatório a qualquer local na memória, mas somente operações sobre blocos de dados. Desta maneira, *flash* tipo NAND opera de forma semelhante a outros dispositivos de armazenamento de dados secundários, tais como discos magnéticos e meios ópticos, e é assim muito utilizada em dispositivos de armazenamento em massa, tais como cartões de memória [1].

Com o crescente uso de memória *flash* em dispositivos portáteis como celulares, *smartphones, tablets* e *pen drives*, cresce também a necessidade de garantir a proteção dos dados armazenados. Muitos arquivos armazenados nestes dispositivos contêm dados pessoais e privados, que não devem ser acessados por pessoas não autorizadas. Este problema de acesso pode acontecer em caso de roubo, perda do dispositivo, inclusive para arquivos que foram removidos. Arquivos removidos não devem ser recuperados por terceiros, pois isto também compromete a privacidade de uma pessoa.

1.1 **Motivação**

Dispositivos de armazenamento de dados são fundamentais em sistemas de computação, e uma regra básica dos projetistas de dispositivos de armazenamento é que os dados devem ser protegidos. Contudo não somente informações sigilosas devem ser protegidas, mas a privacidade do usuário também é muito importante. Com a facilidade de expor em redes públicas informações sigilosas, o usuário deve se proteger de atacantes que vasculham informações que já deveriam ter sido apagadas do meio de armazenamento. Como muitas vezes este apagamento é somente lógico e não físico, informações logicamente removidas podem ser recuperadas através da análise

do meio físico. Assim, para preservar a privacidade do usuário, deve haver a possibilidade de informações pessoais serem imediata e permanentemente apagadas de dispositivos de armazenamento. Este requisito desempenha um papel crítico em todos os sistemas de práticas de gerenciamento de dados [44].

Eliminação segura de dados consiste em suprimir permanentemente os dados digitais de um meio físico de tal modo que os dados se tornem irrecuperáveis [13]. Assim, por exemplo, métodos antiforenses [10] são utilizados para eliminar permanentemente arquivos específicos ou sistemas de arquivos inteiros [57]. Isto pode ser feito através da utilização de uma variedade de métodos que incluem o uso de utilitários de limpeza de disco, de arquivos e desmagnetização/destruição de discos. Para a eliminação lógica de dados, a solução mais comum usa deleção por sobrescrita (*overwriting*) que já foi proposta em diversos artigos [3], [13], [14].

Contudo, isso dá origem a outro problema. O uso de dispositivos com memórias não voláteis aumentou significativamente nos últimos anos, como ilustra a Figura 1, retirada de [34]. Com o advento da Internet das Coisas (IoT), é crescente o número de dispositivos que contêm memória não-volátil sendo usados em situações diárias. Por questões de privacidade dos usuários, garantir que os dados contidos nestas memórias sejam devidamente apagados também se tornou uma preocupação. Quando *pendrives*, celulares, *tablets* e afins são roubados, perdidos ou descartados, os dados continuam armazenados na memória. Mesmo que o usuário tenha o cuidado de excluir os arquivos, os dados permanecem na memória *flash* e ainda podem ser recuperados [59] [60].

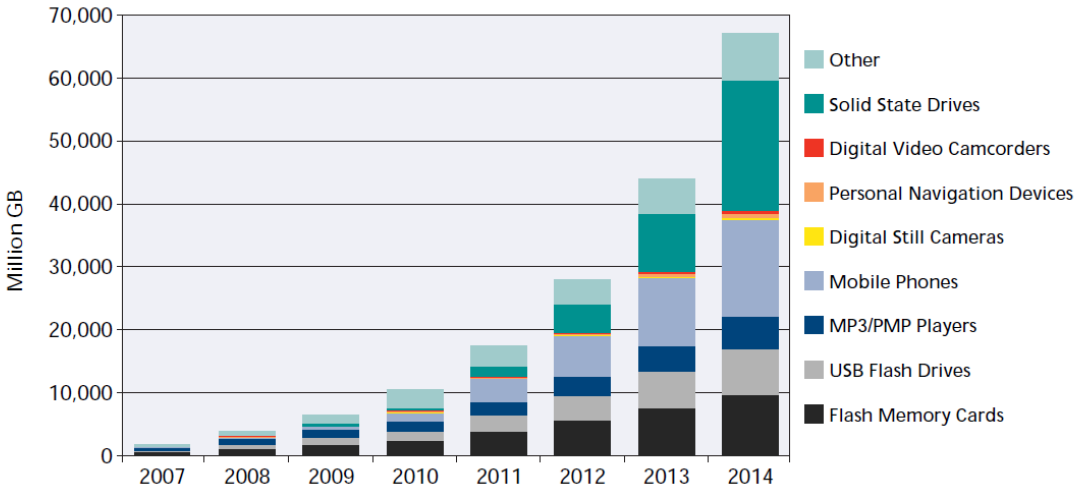


Figura 1 – Demanda por memória não-volátil [34]

Com a popularização de dispositivos móveis na Internet das Coisas, garantir a remoção segura de dados é uma preocupação que atinge tanto empresas e órgão governamentais, por questões de sigilo, como também usuários comuns, por questões de privacidade.

1.2 **Objetivos**

Este trabalho visa estudar, analisar e aperfeiçoar métodos de eliminação segura de arquivos em memória não-volátil do tipo *flash*. Uma eliminação deste tipo é necessária para impedir que

pessoas não autorizadas, utilizando métodos forenses, possam recuperar dados que comprometem a privacidade e a vida particular de uma pessoa.

Especificamente, este trabalho se propõe a:

1. Estudar e analisar métodos existentes de eliminação segura de arquivos em memória não-volátil.
2. Verificar a aplicabilidade dos métodos de eliminação segura nos arquivos de uma memória *flash*.
3. Analisar os métodos aplicáveis a uma memória *flash* do ponto de vista de eficácia e eficiência.
4. Analisar as resistências dos métodos contra possíveis ameaças e ataques.
5. Verificar a possibilidade de melhorar os métodos aplicáveis a uma memória *flash*.
6. Validar estes métodos em diferentes cenários de uso e com arquivos de diversos tamanhos.

1.3 **Organização**

O restante deste documento é organizado da seguinte forma: o Capítulo 2 discute conceitos de segurança de arquivos assim como alguns métodos de recuperação e remoção segura. No Capítulo 3, são apresentadas e discutidas questões sobre memória *flash*, dando a motivação principal para o estudo de eliminação segura em memória não-volátil. Trabalhos relacionados sobre métodos desenvolvidos para eliminar dados em memória *flash* são vistos no Capítulo 4. O Capítulo 5 modela e analisa estes métodos e realiza uma comparação teórica entre eles. No Capítulo 6 descreve-se as características do Simulador, implementado para avaliar o desempenho prático dos métodos estudados. No Capítulo 7 são analisados os desempenhos dos diversos métodos através de experimentos com o Simulador, e o método proposto é discutido e refinado. O Capítulo 8 analisa os diversos métodos de um ponto de vista qualitativo, inclusive descrevendo ameaças aos métodos de remoção. Finalmente, o Capítulo 9 apresenta as conclusões e ideias de prosseguimento do trabalho.

2 SEGURANÇA DE ARQUIVOS

Segurança da Informação pode ser definida como o processo de proteção dos sistemas de informação contra acesso não autorizado ou a alteração das informações, sejam essas informações armazenadas, em processamento ou em trânsito. Também devem ser inclusas nesse processo as proteções relativas a sistemas de arquivos, visando impedir acessos não autorizados e outras ameaças, incluindo todas as medidas para detectar, documentar e combater essas ameaças, incluindo negação de serviço a usuários legítimos [49]. Com a disseminação de sistemas computadorizados, crimes digitais se tornaram uma prática comum, afetando tanto usuários como empresas. Para auxiliar na análise destas ameaças, surgiu a Forense Computacional, que permite compreender quais são os requisitos necessários a uma investigação digital. Nessa Seção serão exploradas as técnicas de forense e antiforense digital, dando mais foco às últimas porque envolvem conceitos de proteção e remoção segura de arquivos.

2.1 **Forense digital**

Forense é o processo de empregar conhecimento científico para coletar, analisar e apresentar evidências perante uma corte. Forense digital, por outro lado, combina elementos legais com ciência da computação para coletar e analisar dados de sistemas computacionais, redes e dispositivos de armazenamento de forma que possam ser usados como evidência [56]. Forense digital coleta dois tipos de dados: voláteis e persistentes. Dados voláteis são os armazenados em memória principal, ou existem em trânsito, e são perdidos quando o computador é desligado ou quando falta energia. Dados persistentes, por outro lado, são os dados mantidos quando o computador é desligado. As técnicas de forense digital utilizam exaustivamente recursos de recuperação de dados persistentes, quer na forma de arquivos, fragmentos de arquivos ou fragmentos de texto que o proprietário do equipamento julgava como excluídos. Entretanto, como todos os métodos computacionais, técnicas de forense digital também podem ser usadas para realizar uma invasão de privacidade de um usuário, pela recuperação não autorizada de dados pessoais registrados em mídias de armazenamento. Em contrapartida, a antiforense digital define métodos de remoção, ocultação e subversão de evidências com o objetivo de mitigar os resultados de análise forenses [10].

2.2 **Antiforense digital**

A antiforense digital pode ser definida como um conjunto de métodos de ocultação e remoção de informações em dispositivos de armazenamento, para proteger a privacidade de pessoas e empresas [57]. Relatos policiais e artigos na área [57] registram com frequência situações onde informações ou imagens sigilosas são divulgadas em redes públicas, prejudicando a reputação de pessoas ou causando perdas financeiras a instituições.

Problemas decorrentes do uso de dados não protegidos de forma eficiente podem ter diversas origens. Uma pesquisa realizada pelo Ponemon Institute (http://www.ponemon.org/), em maio de 2008, indica o registro de roubo ou perda de mais de 637.000 notebooks no período de um ano, exclusivamente em aeroportos [33]. Além disso, é comum verificar, em meios de comunicação, notícias relacionadas a situações onde informações ou imagens sigilosas são divulgadas em redes

públicas, segredos corporativos são compartilhados com a concorrência e o acesso a informações financeiras privadas facilita negócios muito lucrativos.

Uma regra fundamental dos projetistas de dispositivos de armazenamento é que os dados devem ser protegidos. Uma unidade de disco, principal dispositivo de armazenamento em computadores pessoais, é projetada para evitar perdas e danos acidentais aos dados. Todo esforço é realizado com base no pressuposto de que excluir informações de um computador pode ser um evento não usual, entretanto a eficiência destas medidas para proteger e agilizar o acesso de usuários aos danos pode transformar-se em vulnerabilidade, quando explorada por pessoas não autorizadas.

A área de antiforense digital investiga, entre outros métodos, como proteger informações sensíveis para impedir sua divulgação não autorizada e assim proteger a privacidade de pessoas e empresas. Não deve ser usada para ocultar provas e evidências de atos ilícitos mas, como várias outras técnicas computacionais, seu uso correto depende muito mais de aspectos éticos do que aspectos tecnológicos.

Uma das categorias antiforense trata da remoção e esterilização de arquivos, normalmente destruindo os dados logicamente através da reescrita do conteúdo com dados aleatórios (sobrescrita – *overwriting/wipe*). Ao invés de ser sobrescrito, um arquivo também pode ser protegido por criptografia, ou até mesmo escondido por esteganografia [45].

Métodos antiforense são categorizados para tornar a classificação das várias ferramentas e técnicas mais simples. Apesar de certas divergências entre as classificações adotadas por certos autores como Harris [16] e Rogers [47], quatro destas são comuns a todos: ocultação de evidências, destruição de evidências, falsificação de evidências e a eliminação das fontes de evidências.

A ocultação de evidências é o processo de tornar dados difíceis de serem encontrados e ao mesmo tempo mantê-los acessíveis para o uso futuro. Algumas das formas mais comuns de ocultação de dados incluem criptografia e esteganografia. Cada um dos diferentes métodos de ocultação de dados dificulta exames forenses digitais e quando combinados eles podem tornar uma investigação forense quase impossível. Umas das técnicas mais utilizadas para desafiar a computação forense é a criptografia de dados. Isto porque através da utilização de algoritmos de criptografia modernos e de várias técnicas de criptografia, diversos programas disponíveis publicamente tornam os dados virtualmente impossíveis de serem lidos sem a chave designada, ou seja, uma senha convencional escolhida a critério do usuário no momento do processo de cifragem. A maioria dos programas de criptografia tem a capacidade de executar uma séria de funções adicionais que tornam esforços forenses digitais cada vez mais inúteis. Algumas dessas funções incluem a utilização de arquivos como chave criptográfica, a cifragem de volumes inteiros e a negação plausível. A ampla disponibilidade de softwares que contém estas funções colocou o campo de forense digital em uma grande desvantagem. Alguns exemplos de ferramentas de cifragem são o TrueCrypt [55] e o BitLocker Drive Encryption [37]. Outra técnica de ocultação de dados é a esteganografia, onde a informação ou arquivos estão escondidos dentro de outro arquivo na tentativa de ocultar dados, deixando-os à vista de todos. A esteganografia produz dados obscuros, normalmente ocultados dentro de dados visíveis (por exemplo, alguns caracteres

de texto escondidos dentro de uma fotografia digital). Alguns especialistas argumentam que técnicas de esteganografia não são muito utilizadas e, portanto, não devem ser levadas muito a sério. Entretanto, a maioria dos especialistas concorda que esteganografia tem a capacidade de interromper o processo forense quando usada corretamente.

Os métodos usados na destruição de evidências são encarregados de eliminar permanentemente arquivos específicos ou sistemas de arquivos inteiros. Isto pode ser feito através da utilização de uma variedade de métodos que incluem o uso de utilitários de limpeza de disco, de arquivos e desmagnetização/destruição de discos. Utilitários de limpeza de disco usam uma variedade de métodos para substituir os dados existentes nos discos. A eficácia de ferramentas de limpeza de disco como técnica antiforense é muitas vezes contestada, pois alguns acreditam que não são completamente eficazes [13]. Utilitários de limpeza de disco também são criticados porque eles deixam sinais de que o sistema de arquivos foi apagado, o que em alguns casos é inaceitável. Alguns dos utilitários de limpeza de disco amplamente usados incluem o CyberScrub [7] e o KillDisk [26]. Em contrapartida a ferramentas de limpeza de disco, a desmagnetização de disco é um processo pelo qual um campo magnético é aplicado a um dispositivo de suporte digital. O resultado é um dispositivo completamente limpo de todos os dados armazenados anteriormente. A desmagnetização é raramente utilizada como um método de antiforense, apesar de ser o meio mais eficaz para garantir que os dados tenham sido completamente apagados. Isto é atribuído ao custo elevado das máquinas de desmagnetização. A destruição de evidências também pode ser realizada através de métodos criptográficos [15].

O objetivo da falsificação de evidências é confundir, desorientar e desviar o processo de análise forense. A falsificação pode comprometer desde apenas um bit até certa quantidade de bits contidos em um disco, com o intuito de parecer qualquer outra coisa, menos a evidência real [16]. O ofuscamento de rastros abrange uma variedade de técnicas e ferramentas que incluem limpadores de logs*, spoofing*, desinformação, e uso de contas zumbis, entre outros.

2.3 **Considerações**

Meios de armazenamento são largamente utilizados atualmente em sistemas de computação, e por consequência sistemas de arquivos são projetados para evitar perdas e danos acidentais aos dados. Técnicas como um diretório para arquivos reciclados (“lixeira”) e comandos *unerase* estão disponíveis na maioria dos sistemas operacionais para prevenir perda indesejada de informações. A exclusão de ponteiros (índices) é um padrão para agilizar a exclusão e possibilitar a recuperação de um arquivo em disco, e *drives* utilizam técnicas de detecção de erros para impedir que uma leitura resulte em valores incorretos [18]. Entretanto, técnicas forenses digitais podem ser empregadas para recuperar informações que foram logicamente removidas de meios de armazenamento, mas ainda persistem fisicamente no meio. Em muitos casos, porém, estas técnicas podem ser subvertidas para recuperar de forma ilegal os dados pessoais de um usuário. Assim, visando proteger a privacidade e os dados pessoais de um usuário, é necessário que sejam empregadas técnicas de remoção segura de arquivos quando se deseja remover permanentemente dados privados.

3 SISTEMAS DE ARQUIVOS EM MEMÓRIA NÃO-VOLÁTIL

A memória *flash* é um meio de armazenamento não-volátil que consiste em uma série de componentes eletrônicos que armazenam informações [2]. A memória *flash* tem massa e volume físicos muito pequenos, não incorre em penalidades para buscar dados com acesso aleatório, e requer pouca energia para funcionar. Como tal, dispositivos portáteis usam quase exclusivamente memória *flash*. Entretanto, memória *flash* tem diferenças significativas com memória volátil (RAM) e com tecnologias de disco magnético, requerendo *drivers* de software e sistemas de arquivos especiais. A Figura 2 ilustra como a arquitetura de uma memória *flash* é relacionada com aplicativos e sistemas de arquivos.

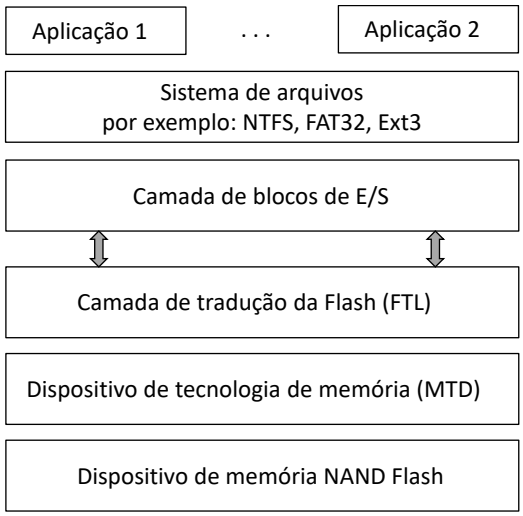


Figura 2 – Arquitetura de uma memória *flash* do tipo NAND

Aplicações fazem operações sobre um sistema de arquivos gerenciado pelo sistema operacional, como por exemplo NTFS, FAT32 ou Ext3. Estas operações sobre arquivos são mapeadas para operações básicas de entrada e saída sobre blocos. Em um disco magnético, estes blocos são chamados de setores, sobre os quais são realizadas as operações de leitura e escrita. Para apagar um setor, basta sobrescrever seu conteúdo através de uma operação de escrita.

Tecnologicamente, memórias *flash* podem ser endereçadas a bloco (NAND) ou a byte (NOR). Sistemas de arquivo são normalmente implementados em memória *flash* NAND. A memória *flash* NAND é dividida em dois níveis de granularidade. O primeiro é chamado nível de unidades de apagamento *(erase blocks* ou *erase units*), que são da ordem de 64KB ou 128 KB [12] no tamanho. Cada unidade de apagamento é dividida em blocos (*blocks* ou *pages*), que são da ordem de 2 KB ou 4 KB de tamanho. Uma unidade de apagamento é a região mínima de apagamento e os blocos são a unidade de operações de leitura e escrita [9]. Não se pode gravar dados em um bloco de memória *flash*, a menos que este bloco tenha sido apagado anteriormente. A operação de apagamento executada em uma unidade de apagamento prepara os *blocos* que ela contém para uma escrita futura.

Apagar memória *flash* causa desgaste físico significativo [36]. Cada apagamento corre o risco de transformar um bloco de apagamento em um bloco ruim, que não pode mais armazenar dados. Blocos de *flash* toleram entre 10^4 – 10^5 apagamentos antes que se tornem blocos não confiáveis

[6]. Para promover uma vida mais longa do dispositivo, apagamentos devem ser nivelados uniformemente sobre os blocos de apagamento.

Para atender estas características, um sistema de armazenamento *flash* NAND normalmente consiste de duas camadas, a FTL (Flash Translation Layer) e a MTD (Memory Technology Device), como ilustrado na Figura 2. A função principal da FTL é redirecionar endereços lógicos do sistema de arquivos do sistema operacional para endereços físicos em *flash* NAND, utilizando uma tabela de mapeamento. FTL também provê componentes úteis como *garbage collector* e *wear-level* para otimizar o espaço utilizado e manter o mesmo nível de desgaste para cada bloco na *flash* NAND. A camada MTD implementa funções primitivas na memória *flash*, assim como, funções de escrever, ler e apagar.

As especificações da FTL descritas nas seções seguintes foram retiradas de [19].

3.1 **Camada de Tradução da *Flash* (Flash Translation Layer)**

Um sistema operacional baseado em disco executa entrada e saída em pedaços estruturados chamados de blocos. Dispositivos de bloco incluem todas as unidades de disco e outros dispositivos de armazenamento em massa no computador. A camada FTL emula um dispositivo de bloco. A mídia *flash* aparece como uma matriz contígua de blocos de armazenamento numerados de zero a menos um que o número total de blocos (Figura 3).

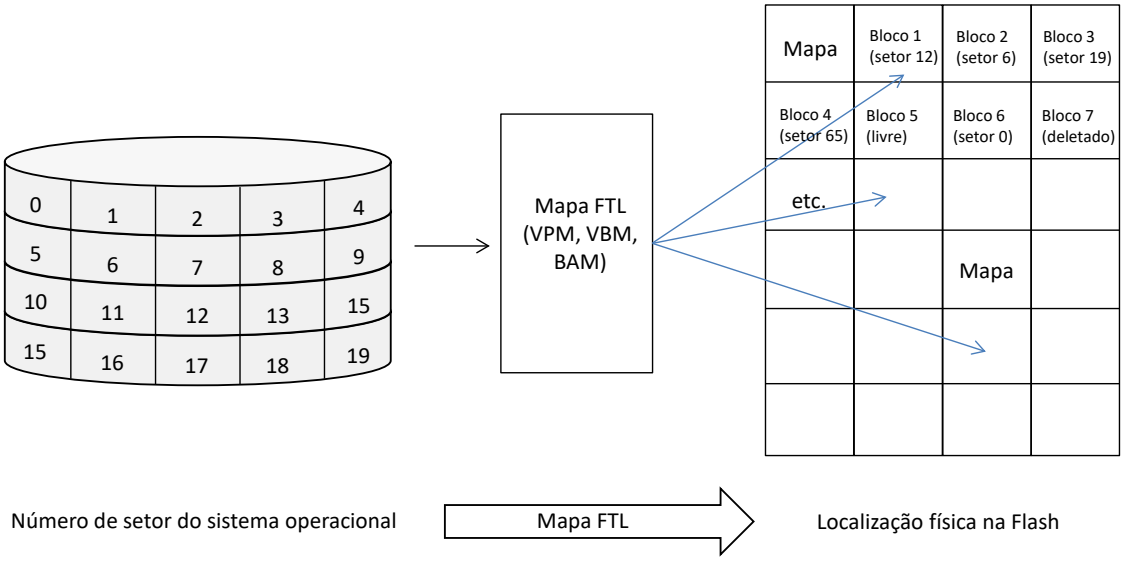


Figura 3 – Mapeamento de blocos lógicos (virtuais) para blocos físicos

FTL é uma camada de tradução entre o sistema de arquivos e a memória *flash*. FTL mapeia os dados para a localização física na qual os dados devem ser escritos. Isso permite que o sistema de arquivos do sistema operacional tratar da *flash* como qualquer outro dispositivo de armazenamento em bloco e ignorar as características do dispositivo de flash. Em teoria FTL recebe os dados do sistema de arquivos e escreve no local especificado (setor). Entretanto na realidade, FTL coloca os dados em uma localização livre/apagada na memória *flash* e regista a localização real dos dados, e também eliminando o bloco que anteriormente continha os dados do bloco (se

qualquer). Então, quando o sistema de arquivos pede os dados que foram escritos, a FTL encontra e lê os dados apropriados.

3.1.1 Unidade de Apagamento

FTL divide a mídia *flash* em uma ou mais unidades de apagamento (*Erase Units)* de igual tamanho (Figura 4). O tamanho de uma unidade de escrita está determinado durante a formatação da FTL, mas também é dependente do tamanho do bloco de apagamento (*Erase Block)* do dispositivo *flash*. Uma unidade de apagamento (*Erase Unit*) consiste em uma ou mais zonas de apagamento (*Erase Zone*) contíguas. Uma zona de apagamento é a menor área contígua que pode ser apagada em uma única operação de apagamento. Uma zona de apagamento é o mesmo que um bloco de apagamento (*Erase Block*) de um *chip flash* (tipicamente 64 KByte em 2007).

Uma Unidade de Apagamento é dividida igualmente em um ou mais blocos de leitura e escrita em tamanho igual. Cada bloco de leitura/escrita tem o mesmo tamanho de um bloco Virtual utilizado pelo sistema de arquivos. Um Bloco Virtual é o mesmo que um setor de dados do sistema operacional (tipicamente 512 bytes).

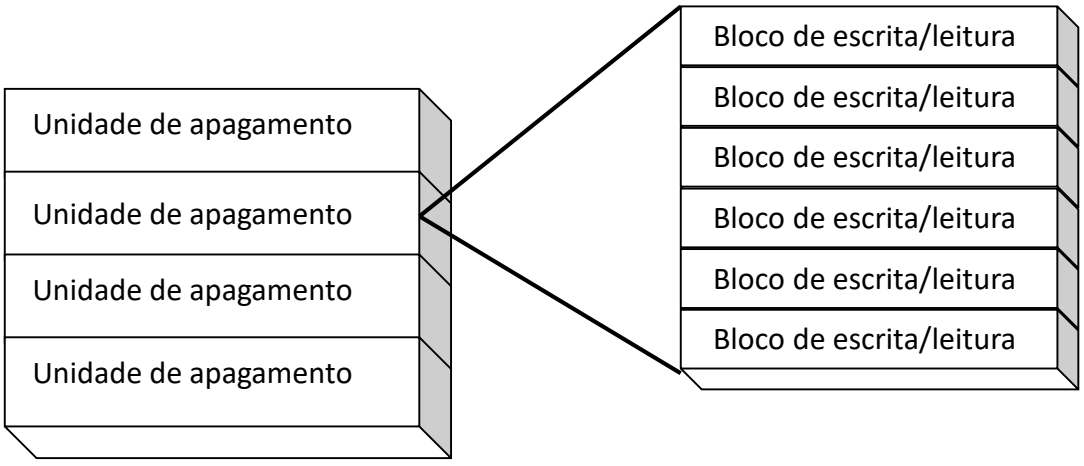


Figura 4 – Estrutura das Unidades de Apagamento e Blocos

3.1.2 Cabeçalho da Unidade de Apagamento

Cada Unidade de Apagamento possui um Cabeçalho da Unidade de Apagamento (EUH, *Erase Unit Header*), localizado no deslocamento 0. O EUH contém diversas informações sobre a partição FTL, sendo que a maioria dos campos é usada globalmente, e só dois são específicos da Unidade de Apagamento. Assim, todos os EUH são idênticos, exceto pelos campos EraseCount e LogicalEUN.

A Tabela 1 mostra o Cabeçalho da Unidade de Apagamento.

Tabela 1 – Cabeçalho da Unidade de Apagamento

Offset	Field	Size	Description	Example: 4-Meg F008-based Card HEX Comments
0	LinkTargetTuple	5	PCMCIA Link Target tuple	13 03 43 49 53 (..CIS)
5	DataOrgazinationTuple	10	PCMCIA Data Organization tuple [43]	46 39 00 46 54 4C 31 30 30 00 (F9.FTL100.)
15	NumTransderUnits	1	Number of transfer units in partition	01
16	Reserved	4	Reserved	xx xx
20	LogicalEUN	2	Logical Erase Unit Number of this block	FF FF
22	BlockSize	1	Size of a Read/Write Block	09h 2^9h = 200h (152 dec)
23	EraseUnitSize	1	Size of an Erase Unit	11h 2^11h = 20000h (128K dec)
24	FirstPhysicalEUN	2	Physical Erase Unit where FTL partition begins	00 00
26	NumEraseUnits	2	Total number of Erase Units in partition	20 00 (0020)
28	FormattedSize	4	Total formatted size of partition	00 10 3C 00 (003C10000)
32	FirstVMAddress	4	First virtual address physically mapped in VBM page on media	00 00 01 00 (00100000)
36	NumVMPages	2	Total number of VBM pages	3D 00 (003D)
38	Flags	1	Special bit-mapped flags	00
39	Code	1	Code designation EDAC type	FF None
40	SerialNumber	4	Partition serial Number	00 00 00 00
44	AltEUHOffset	4	Offset of alternative EUH	00 00 00 00
48	BAMOffset	4	Offset of BAM from start of EUH	44 00 00 00 (00000044)
52	Reserved	12	Reserved	xx...

3.1.3 Informação da Alocação de Blocos

Cada Unidade de Apagamento mantém informações de alocação sobre cada bloco de leitura/escrita na Unidade. Cada bloco de leitura/escrita tem um valor de 4 bytes informando o seu estado atual. O bloco de leitura/escrita em uma Unidade de Apagamento ou é excluído, defeituoso, livre ou alocado (*deleted, bad, free, allocated*). Blocos de leitura/escrita armazenam de quatro tipos de dados: Estruturas de Controle, Blocos Virtuais de Dados, Mapeamento de Blocos Virtuais e Páginas de Substituição (FTL Control Structures, Virtual Block Data, Virtual Block Map Pages e Replacement Pages).

3.1.4 Mapa da Alocação de Blocos

De acordo com a especificação FTL, a informação de atribuição de blocos pode ser armazenada em duas formas diferentes. Um método consiste em armazenar a alocação em áreas escondidas próximas ou relacionadas com o bloco de leitura/escrita a que se refere. O outro método é armazenar as informações de alocação para todos os blocos de leitura/escrita na Unidade de Apagamento em um vetor chamado Block Allocation Map (BAM). Esta é a técnica mais comum usada pelas soluções FTL disponíveis comercialmente.

Tabela 2 – Informações do Bloco de Alocação

32-Bit BAM Entry	Significado	Descrição
0xFFFFFFFF	Livre	Bloco de leitura/escrita está disponível para escrita.
0xFFFFFFFE ou 0x00000000	Deletado	Dados no bloco são inválidos. Esse bloco precisa ser apagado antes de ser usado de novo. O valor 0xFFFFFFFE indica que a operação de escrita foi interrompida antes de sua completude. O valor 0x00000000 indica que os dados no bloco estão obsoletos.
0x00000070	“Bad”	O bloco é inutilizável e não pode ser escrito ou lido.
0x00000030	Controle	Esse bloco contem estruturas de controle de FTL
0xXXXXXX40	Dados ou Map Page	Contem dados ou uma Virtual Map Page.
0xXXXXXX60	Replacement Map Page	Replacement Page for a Virtual Map Page

Um número de uma entrada é interpretado de acordo com a Tabela 2. Tem-se assim, por exemplo:

- 0x00003240: página de dados, com endereço lógico 0x3200
- 0xFFFFFC40: página de mapa, sendo página VBM 2
- 0x001B2440: página de dados, com endereço de página 0x1B2400
- 0xFFFFFA40: página de mapa, sendo página VBM 3

- 0xFFFFFE40: página de mapa, sendo página VBM 1
- 0xFFFFFE60: página de substituição, sendo página de substituição VBM 1

A Figura 5 mostra um exemplo de um Mapa da Alocação de Blocos.

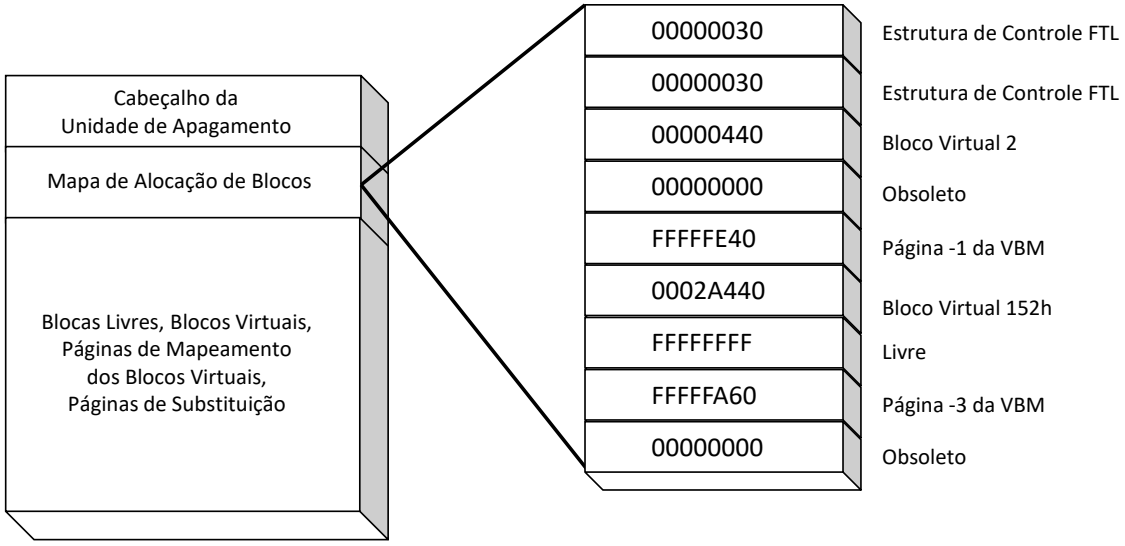


Figura 5 – Mapa de alocação de blocos

3.1.5 Verificação da Partição FTL

Existem duas maneiras de identificar uma partição de FTL. Uma partição FTL pode ser identificada através da leitura da Estrutura de Informações sobre o Cartão (Card Information Structure – CIS) ou através de uma varredura na mídia de armazenamento procurando por estruturas de dados FTL.

Se FTL usa toda a mídia, a CIS não é necessária para conter informações de partição. Uma partição FTL pode ser reconhecida se um Cabeçalho de Unidade de Apagamento é encontrado no primeiro megabyte da mídia e as informações no cabeçalho são válidas. A pesquisa procura pelo FTL Data Organization Tuple (ver Tabela 1, deslocamento 5). Se essa tupla não é encontrada dentro do primeiro megabyte da área de partição, a mídia não utiliza um armazenamento de dados FTL. Se a tupla for encontrada, o restante das entradas no Cabeçalho de Unidade de Apagamento deve ser validado.

Um mapa dinâmico da tradução lógica para física vai ser construído durante este processo de validação. A FTL vai ler cada EUH na mídia começando com a unidade descrita pelo campo FirstPhysicalEUN. Se o EUH não for encontrado, a unidade é tratada como uma unidade de transferência. Se existirem duas unidades de Apagamento com o mesmo LogicalEUN, uma das duas unidade de Apagamento vai ser tratada como uma unidade de transferência. Depois que todos os EUHs foram lidos, o número total de unidades com LogicalEUNs distintas e não negativas deve ser igual NumEraseUnits menos NumTransferUnits.

3.1.6 Mapeamento de Blocos Virtuais para Blocos Lógicos

FTL executa traduções de mapeamento virtual para lógico através do uso de um Virtual Block Map (VBM). O VBM consiste em uma ou mais páginas, cada uma do mesmo tamanho que os blocos virtuais. Cada página VBM é uma matriz de 32 entradas. Cada entrada aponta para um endereço lógico na mídia onde o bloco de dados virtual correspondente está localizado. Entradas VBM são armazenadas em ordem *little-endian* na mídia. FTL usa o número de bloco virtual do sistema de arquivos do sistema operacional como o índice em uma página VBM. O VBM é construído a partir dos endereços virtuais dos blocos de dados virtuais.

O tamanho do VBM pode ser calculado dividindo o tamanho da mídia (FormattedSize) pelo tamanho do bloco virtual (BlockSize) e multiplicando o resultado pelo tamanho de uma entrada VBM (32 bits). O número de páginas (NumVMPages) necessários para o VBM é encontrado tomando o tamanho do VBM, dividindo-o pelo BlockSize e arredondando para cima o número inteiro para o mais próximo. Cada página VBM é o tamanho de um bloco de leitura/escrita. Espaço para toda a VBM é sempre reservado na mídia, independente do fato do VBM estar ou não na própria mídia. No momento da formatação da mídia, FTL indica no Campo FirstVMAddress do Cabeçalho da Unidade de Apagamento exatamente quanto do VBM é realmente mantida na mídia.

3.1.7 Mapeamento de Páginas Virtuais – localizando as páginas do VBM

Se qualquer parte do VBM é mantida na mídia, FTL deve rastrear e localizar essas páginas. A Página do Mapa Virtual (Virtual Page Map – VPM) desempenha uma função semelhante à do VBM. Em vez de mapear a localização de blocos de dados virtuais, cada entrada na VPM mapeia a localização de uma página VBM. A especificação FTL não define uma maneira de manter o VPM na mídia e, portanto, normalmente deve ser construído quando a mídia é acessada pela primeira vez. Isto é conseguido varrendo a mídia por entradas VBM e construindo uma tabela na memória RAM do sistema.

3.1.8 Mapeamento Lógico para Físico

Todos os endereços armazenados nas matrizes VBM (Virtual Block Map) e VPM (Virtual Page Map) são endereços lógicos. O endereço lógico aponta para uma localização na mídia. O campo LogicalEUN indica o número lógico de cada bloco. FTL mapeia as LogicalEUN a um PhysicalEUN fazendo a varredura da mídia e gravando as relações em uma matriz no momento da inserção de inicialização / inserção do cartão. Essas relações são usadas para traduzir entre endereços físicos e lógicos. Um endereço lógico pode ser tratado como tendo duas partes distintas. A primeira, que consiste em os bits mais significativos, refere-se ao LogicalEUN. A segunda parte, que consiste nos bits menos significativos, é um deslocamento dentro da Unidade de Apagamento. O número de bits em cada parte depende do tamanho da Unidade de Apagamento.

3.1.9 Leitura

A FTL processa operações de leitura de blocos requisitadas pelo sistema operacional. Cada bloco Virtual é identificado por um número pelo sistema operacional. Este número é utilizado como um índice para o VBM (Virtual Block Map), para obter o endereço lógico do bloco a ser lido. Para realizar a leitura, a FTL traduz o endereço lógico para um endereço físico. Se a entrada VBM para um bloco é 0xFFFFFFFF, então o bloco Virtual não existe e um *buffer* de zeros é retornado para o sistema operacional. Se a entrada é 0x00000000, então o endereço lógico pode estar localizado em uma Página de Substituição. Se a Página de Substituição não existe ou sua entrada é 0x00000000, então o bloco Virtual não existe e um *buffer* de zeros é retornado para o sistema operacional.

3.1.10 Escrita

Escrever em dados de blocos virtuais é uma das partes mais complexas do FTL. Quando o sistema requisita a FTL para realizar uma operação de escrita, a FTL precisa encontrar um bloco livre para realizar tal ação. Caso nenhum seja encontrado, a Recuperação da Unidade de Apagamento deve ser efetuada. Se um bloco com espaço desocupado (marcado como *free*, livre) for encontrado, a sua entrada no *Block Allocation Map* (BAM) é marcada com 0xFFFFFFF, para indicar que uma escrita está ocorrendo neste bloco. Entretanto, se a escrita for interrompida a FTL será capaz de marcar aquele bloco como deletado. Após a conclusão da operação de escrita, a entrada BAM é atualizada para o endereço virtual do bloco Virtual. Os dados do Bloco Virtual são então escritos no o bloco de escrita / leitura. O VBM é atualizado para apontar para a nova área atribuída para este bloco Virtual. Se o bloco sendo escrito está substituindo um bloco já existente, então sua entrada antiga no BAM é marcada com 0x00000000.

3.1.11 Recuperação de uma Unidade de Apagamento

À medida que a memória é usada, eventualmente não haverá mais espaço disponível. Blocos de leitura/escrita que foram marcados como excluídos ou substituídos só podem ser reutilizados depois de ser retornado ao estado apagado (*erased*). Devido à tecnologia utilizada em *flash*, todos os blocos de leitura/escrita na mesma Unidade de Apagamento devem ser apagados ao mesmo tempo. Entretanto, não é usual que todos os blocos de leitura/escrita na Unidade de Apagamento sejam marcados como eliminado ou substituído ao mesmo tempo. O processo de recuperação da unidade (*reclaim*) preserva os dados em blocos de leitura/escrita válidos enquanto recupera os blocos excluídos na mesma Unidade de Apagamento.

A unidade de recuperação precisa que a mídia tenha pelo menos uma Unidade de Transferência utilizável. A Unidade de Transferência é uma Unidade de Apagamento em um estado de apagamento (*erased*), exceto por sua EUH (*Erase Unit Header*, Cabeçalho da Unidade de Apagamento). Unidades de Transferência são utilizadas para armazenar dados válidos de uma Unidade de Apagamento sendo apagada. Após a localização de uma unidade de transferência devidamente preparada, FTL define a LogicalEUN da unidade para 0x7FFF para indicar que uma recuperação desta unidade está em andamento. A unidade de transferência devidamente preparada é aquela que tem as áreas globais do EUH inicializadas e que tem regiões apagadas para

os Blocos Virtuais de dados, Páginas VBM, Páginas de Substituição e o BAM. Após a conclusão do procedimento de recuperação da unidade, a LogicalEUN será mudada para o LogicalEUN da Unidade de Apagamento recuperada (*recovered*). A antiga Unidade de Apagamento é apagada e reformatada como uma unidade de transferência.

3.2 **Dispositivo de Tecnologia de Memória (*Memory Technology Device*)**

Memória *flash* apenas permite dois estados: apagados e não apagados. No estado de apagado, um byte pode ser todos uns (0xFF) ou todos zeros (0x00), dependendo da tecnologia do dispositivo *flash*. Um bit só pode ser escrito quando a mídia está em estado de apagado. Depois que ele é escrito, o bit é considerado usado e imutável. Para retornar o bit para o seu estado de apagado, um bloco significativamente maior da mídia *flash*, chamado de Zona de Apagamento (também conhecido como um bloco de apagamento) deve ser apagada. A tecnologia Flash não permite a alternância de bits ou bytes individuais de um estado não-apagados de volta a um estado apagado. A FTL torna estes detalhes transparentes para o sistema de arquivos e remapeia os dados passados para ela escrevendo em áreas de dados não utilizadas na mídia *flash*. Isto apresenta a ilusão para o sistema operacional que um bloco de dados é simplesmente substituído quando ele é modificado quando, na realidade, os novos dados foram gravados em outro lugar na mídia. FTL também cuida de recuperar os blocos de dados descartados para reutilização.

Embora haja muitos tipos e fabricantes de memória *flash*, o tipo mais comum é conhecido como *flash* NAND. NAND *flash*, tal como o disponível da Intel Corporation, opera da seguinte maneira: estado de apagado é 1, estado programado é 0, um 0 não pode ser alterado novamente para a 1, exceto por um apagamento. A Figura 6 ilustra este efeito.

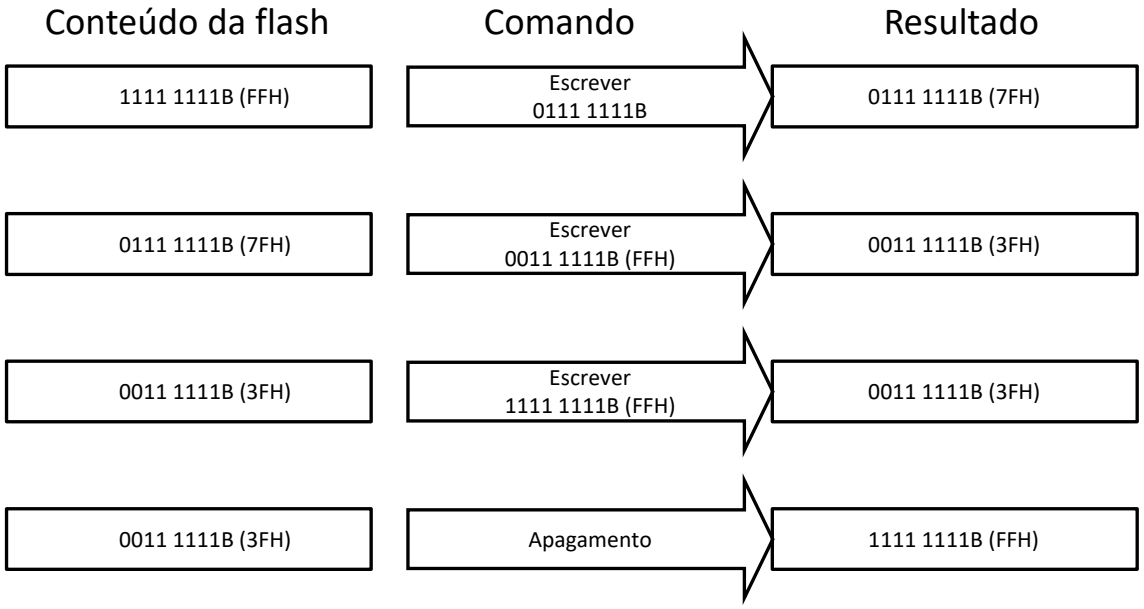


Figura 6 – Operações de escrita e apagamento

Um *chip* de memória *flash* NAND consiste de várias unidades, e cada unidade é composta de um número fixo de blocos. Um bloco é a unidade básica para a operação de leitura/escrita, enquanto uma unidade é a quantidade mínima para uma operação de apagamento. Um chip de memória

flash típico suporta três tipos de operações: ler, escrever e apagar. O desempenho destas três operações é bastante diferente, como mostrado na Tabela 3. Uma operação de apagamento leva muito mais tempo do que uma escrita ou de leitura. Com a realização de operações de escrita em um chip de memória *flash*, o espaço livre diminui e operações de coleta de lixo são invocadas para recuperar algum espaço livre para reutilização. A operação de coleta de lixo pode incluir um número de funções de leitura, escrita, e as operações de apagamento. Uma vez que a coleta de lixo não pode ser interrompida, uma solicitação de gravação pendente poderá ser bloqueada e o seu tempo de resposta será significativamente afetada pela latência da coleta de lixo. A Tabela 3 ilustra tempos típicos para memória *flash* NAND.

Tabela 3 – Especificações de Memória *flash* NAND [58]

Características	**Toshiba SLC NAND Flash**	**Toshiba MLC NAND Flash**
Leitura	30 μs	250 μs
Escrita	250 μs	2700 μs
Apagamento	3ms	4ms
Tamanho de bloco	4.328 Bytes	8.568 Bytes
Tamanho de unidade	270,5 KBytes	1.606,5 KBytes

3.3 **Remoção segura de arquivos**

A memória *flash* é utilizada universalmente em dispositivos portáteis. No entanto, a forma como os sistemas modernos usam memória *flash* tem um sério inconveniente, não garante exclusão de dados armazenados. Para o usuário, dados parecem ter sido excluídos do sistema de arquivos, mas na realidade continuam a ser acessíveis após eliminação. Desta maneira, remoção de arquivos em memórias *flash* é um problema quando se deseja a destruição física dos dados [23].

Resumo das características de memória *flash*:

- Memórias *flash* podem ser endereçadas a bloco (NAND) ou a byte (NOR). Sistemas de arquivo são normalmente implementados em memória *flash* NAND.
- Memória *flash* tem três operações: *Read*, *Write* e *Erase*. Antes de uma operação de escrita (*write*) deve ser feita uma operação de apagamento (*erase*).
- Escrita e leitura são feitas sobre blocos (*blocks*), com tamanho típico de poucos KB (por exemplo, 4KB). Já apagamento é feito sobre blocos maiores (blocos de apagamento ou unidades de apagamento), da ordem de dezenas de KB (por exemplo, 64KB).
- Um apagamento somente é realizado quando todos os seus blocos de dados não pertencem mais a nenhum arquivo. Assim, por exemplo, um bloco de 64KB só é apagado quando seus 16 blocos de 4KB não estiverem mais em uso.
- Para evitar desgaste prematuro (*wear off*) o controlador de uma *flash* distribui as escritas uniformemente por toda a memória. Para fazer isso, ele mapeia os blocos físicos em blocos lógicos, de tal maneira que o sistema operacional e o sistema de arquivos somente precisam tratar com blocos lógicos.

- Devido a este mapeamento, uma operação de sobrescrita, que iria apagar os dados em um disco magnético, não consegue apagar os dados de uma memória *flash*. A sobrescrita simplesmente seria realizada em outro setor físico, mas que seria mapeado para o setor lógico onde a sobrescrita seria feita.

Devido a estas características, a tarefa de eliminação segura de arquivos em memórias *flash* é muito complexa. Eliminação segura é a operação de esterilização de dados num meio de armazenamento, de modo que o acesso aos dados já não é possível no meio de armazenamento [11]. Isto está em contraste com o padrão de eliminação utilizado pelos sistemas operacionais, onde são excluídos somente os metadados de um descritor de arquivo, liberando assim o local de armazenamento dos dados, que pode ser reutilizado. O tempo entre a marcação de dados como excluídos e sua real (segura) eliminação é chamado de latência de eliminação. E usa-se o termo "apagamento seguro garantido" para denotar exclusão segura com um limite superior fixo e finito sobre a latência de eliminação de todos os dados.

Em mídias de armazenamento magnético, exclusão segura de dados é implementada substituindo o conteúdo de um arquivo com informações não-sensíveis [46], ou modificando o sistema de arquivos para substituir automaticamente qualquer setor descartado [3]. No entanto, a memória *flash* não pode executar atualizações no mesmo setor de dados (ou seja, *overwrite*) [9]; em vez disso executa apagamento em blocos de apagamento, que têm uma granularidade maior do que as operações de leitura/escrita. Um único bloco de apagamento pode armazenar dados de arquivos diferentes, por isso só podem ser apagados quando todos os dados no bloco de apagamento são marcados como excluídos ou quando os dados são replicados em outros lugares. Além disso, a memória *flash* degrada a cada apagamento, então apagamentos frequentes encurtam a vida útil do dispositivo. Portanto, a solução simples de apagar qualquer bloco de apagamento que contém dados que foram excluídos é muito caro em relação ao tempo e ao desgaste do dispositivo [51].

Pelas características descritas acima, os métodos tradicionais de sobrescrita utilizados em meios magnéticos não são efetivos em memória *flash*. Por isso é preciso garantir uma maneira de remover os dados de uma memória *flash* de forma segura, impossibilitando que eles sejam recuperados.

4 MÉTODOS PARA REMOÇÃO SEGURA

Este capítulo analisa métodos de apagamento seguros para memória *flash* e os principais métodos para remoção de dados em memória *flash* que foram desenvolvidos até esse momento. O problema causado por dados que permanecem em memória foi descoberto pela primeira vez em meio magnético [13]. Mesmo se a informação é substituída várias vezes em discos e fitas, ainda pode ser possível extrair os dados iniciais. Isto levou ao desenvolvimento de métodos especiais para a remoção de informações confidenciais de forma confiável para a mídia magnética.

P. Gutmann propôs uma técnica que leva 35 passos síncronos para desmagnetizar a mídia magnética [13]. A técnica proposta foi implementada em ferramentas de usuário e em um sistema de arquivos Linux [3]. Em adição ao método de substituição, existem métodos que modificam os sistemas de arquivos para suportar eliminação segura. N. Joulov projetou Purgefs, uma extensão do sistema de arquivos que substitui transparentemente arquivos conforme os mesmos são deletados [24]. Purgefs pode ser adicionado automaticamente à maioria dos sistemas de arquivos existentes, como os ext2, vfat, msdosfs, ranmfs, NFS e Base0fs. Com esta portabilidade e transparência, os usuários podem convenientemente excluir dados. Além disso, Boneh modifica um sistema de arquivos para criptografar dados em um disco, e os dados podem ser seguramente excluídos ao excluir a chave de criptografia correspondente [5]. Para adicionar contribuições a estes trabalhos, Marley trata de como e onde alguns arquivos são criados e como removê-los com segurança de um sistema [31].

Além de artigos acima mencionados, muitos sistemas para exclusão segura de dados no disco rígido são patenteados. A maior parte destas patentes baseia-se na sobrescrita (*overwriting*). Artigos [22] [25] desenvolvem sistemas e métodos para o apagamento seguro de arquivos, sobrescrevendo o seu conteúdo com padrões binários projetados para impedir qualquer recuperação.

Com o crescente uso da memória *flash*, as preocupações sobre exclusão segura de memória *flash* também são aumentadas. No entanto, devido à característica de memória *flash*, que uma operação de apagamento pode ser realizada a cerca de 10^4 a 10^5 vezes em cada bloco [6], técnicas que são geralmente utilizados na unidade de disco rígido não são aplicáveis para a memória *flash*. Devido a essas limitações, existem alguns métodos existentes para a eliminação segura de memória *flash*.

Em 2008 foram propostos dois métodos de apagamento seguros para memória *flash*. O primeiro método é substituir todos os dados de um arquivo [51]. O segundo é criptografar os dados do arquivo e usar o primeiro método para excluir uma chave de criptografia com segurança [29]. Outros métodos utilizados, como o de [28] e o de [17] empregam estes mesmos métodos, entretanto com algumas variações, como é visto nas seções a seguir.

4.1 **Eliminação segura pelas normas do NIST**

Esforços históricos para limpar dados de meios magnéticos se concentraram em métodos que eram eficazes para este tipo de mídia, amplamente utilizada como meio de armazenamento e implementada de forma relativamente semelhante entre todos os fornecedores e modelos. Tecnologias alternativas, tais como dispositivos de armazenamento baseados em memória *flash* ou discos de estado sólido (Solid State Drives, SSDs), também se tornaram predominantes devido à queda nos custos, maior desempenho e resistência a choques. Discos SSD já começaram a alterar a norma na tecnologia de armazenamento, e, pelo menos a partir de uma perspectiva de remoção segura, a mudança é revolucionária (em oposição a evolucionária). Desmagnetização, uma maneira fundamental para higienizar meios magnéticos, já não se aplica na maioria dos casos para dispositivos baseados em memória *flash*. Mudanças evolutivas em meios magnéticos também terão impactos potenciais sobre a sanitização. Novas tecnologias de armazenamento, e até mesmo variações de armazenamento magnético, que são dramaticamente diferentes de meios magnéticos legados, claramente exigirão pesquisa em remoção segura, assim como uma revisão dos procedimentos de remoção segura para garantir a eficácia [40].

Para dispositivos de armazenamento contendo meios magnéticos, uma única passagem de sobrescrita com um padrão fixo, como zeros binários, normalmente dificulta a recuperação de dados, mesmo que o estado da arte das técnicas de laboratório sejam aplicadas para tentar recuperar os dados [40]. Os usuários que se acostumaram a depender de técnicas de sobrescrita em suporte magnético e que têm continuado a aplicar estas mesmas técnicas conforme os tipos de mídia evoluíram (como dispositivos baseados em memória *flash*) podem estar expondo seus dados ao risco de divulgação não intencional.

Técnicas destrutivas para alguns tipos de mídia podem se tornar mais difíceis ou impossíveis de aplicar no futuro. Técnicas tradicionais, como desmagnetização (para mídia magnética) tornam-se mais complicadas, porque algumas variações emergentes das tecnologias de gravação magnética passaram a incorporar mídia com maior coercitividade (força magnética). Como resultado, desmagnetizadores existentes podem não ter força suficiente para desmagnetizar efetivamente essas mídias. A Aplicação de técnicas destrutivas para a mídia de armazenamento eletrônico (por exemplo, memória *flash*) também está se tornando mais desafiadora, como o tamanho de partícula necessário para as técnicas de trituração comumente aplicadas desce proporcionalmente aos aumentos de densidade de armazenamento de memória *flash*.

Como descrevem as normas do NIST [40], Apagamento Criptográfico (*Cryptographic Erase* – CE) é uma técnica emergente de limpeza de dados que pode ser utilizada em certas situações, quando os dados são codificados e armazenado em mídia. Com CE, a limpeza da mídia é realizada através da limpeza das chaves criptográficas utilizadas para criptografar os dados, ao contrário de limpar os dados locais de armazenamento em mídia contendo os dados criptografados em si. Técnicas CE são tipicamente capazes de limpeza de mídia muito rapidamente e poderiam realizar uma limpeza parcial, onde apenas um subconjunto da mídia de armazenamento deve ser limpo. Limpeza parcial, por vezes referida como limpeza seletiva, tem aplicações potenciais em computação em nuvem e dispositivos móveis. Entretanto, o uso operacional do CE hoje apresenta alguns desafios.

Em alguns casos, pode ser difícil verificar que o CE tenha efetivamente limpado esses meios. Se a verificação não pode ser executada, devem ser utilizados métodos alternativos que podem ser verificados, ou então usar CE em combinação com uma técnica de limpeza que possa ser verificada.

Apagamento Criptográfico aproveita o fato dos dados a serem removidos estarem criptografados, permitindo que a limpeza seja feita somente na chave de criptografia utilizada. Isso deixa apenas o texto cifrado restante na mídia, efetivamente limpando os dados, impedindo o acesso à leitura. Sem a chave de criptografia usada para criptografar os dados, esses dados são irrecuperáveis. O nível de esforço necessário para decodificar essas informações sem a chave de criptografia, então, é o menor entre a força da chave criptográfica ou a força do algoritmo criptográfico e modo de operação usado para criptografar os dados.

Se uma criptografia forte é usada, a limpeza dos dados alvo é reduzida à limpeza da chave(s) de criptografia usada para criptografar os dados alvos. Assim, com CE, a limpeza dos dados pode ser realizada com alta confiabilidade e muito mais rápida do que com outras técnicas de limpeza.

Tipicamente, CE pode ser executado em uma fração de um segundo. Isto é especialmente importante conforme dispositivos de armazenamento ficam maiores, resultando em outros métodos de limpeza de dados levarem mais tempo. CE também pode ser utilizado como um suplemento ou adição a outras abordagens de limpeza.

Nas recomendações do NIST, não deve ser usado CE para limpar uma mídia, se a criptografia foi ativada depois que dados sensíveis foram armazenados no dispositivo sem que esse tenha sido limpo em primeiro lugar. Também não deve ser usado CE, se não se sabe ou se desconhece se os dados sensíveis foram armazenados no dispositivo sem que esse tenha sido limpo antes da criptografia.

Por outro lado, ainda de acordo com as recomendações do NIST, pode se considerar o uso de CE, quando todos os dados destinados a CE são criptografados antes de serem armazenados na mídia (incluindo os dados temporários, bem como cópias virtuais). Também pode ser considerado o uso de CE, quando se sabe a localização(s) na mídia onde a chave de criptografia é armazenada e se pode limpar essas áreas, utilizando uma técnica de limpeza de dados específica e adequada à mídia, garantindo que a localização real na mídia em que a chave é armazenada é utilizada (e não uma localização virtual). Para o uso de CE, deve-se garantir que todas as cópias das chaves de criptografia usadas para criptografar os dados de destino são limpas.

Se a chave de cifragem existe fora do dispositivo de armazenamento (tipicamente, devido a cópias de reserva ou de custódia), há uma possibilidade de que a chave pode ser utilizada no futuro, para recuperar os dados armazenados na mídia cifrada. Assim CE só deve ser usado como um método de limpeza quando a organização tem a confiança de que as chaves de criptografia usadas para criptografar os dados de destino foram devidamente protegidas. Tais garantias podem ser difíceis de obter com módulos de criptografia de software, tais como aqueles usados com soluções de criptografia completa de disco baseadas em software, uma vez que estes produtos geralmente armazenam chaves criptográficas no sistema de arquivos ou em outros locais na mídia acessíveis

ao software. Embora possa haver situações em que o uso do CE com módulos de criptografia de software é apropriado e vantajoso, como realizar um rápido apagamento remoto em um dispositivo móvel perdido, a menos que a organização tem a confiança tanto na proteção das chaves de criptografia, e a destruição de todas as cópias dessas chaves no processo de desinfecção, CE deve ser utilizado em combinação com outro método adequado de limpeza.

Limpeza usando CE não deve ser confiável em dispositivos que tenham sido submetidos a cópias de reserva ou de custódia da chave(s) a menos que a organização tem um alto nível de confiança sobre como e onde as chaves foram armazenadas e gerenciadas fora do dispositivo.

De uma forma geral, as recomendações do NIST [40] consideram três categorias principais de classificação de mídias de armazenamento quanto a sua segurança: baixa, moderada e alta. Para um eventual reaproveitamento da mídia (ou não), três ações são consideradas: limpeza, exclusão e destruição (*clean, purge, destroy*). A Figura 7 ilustra as ações a serem tomadas em função da classificação de segurança.

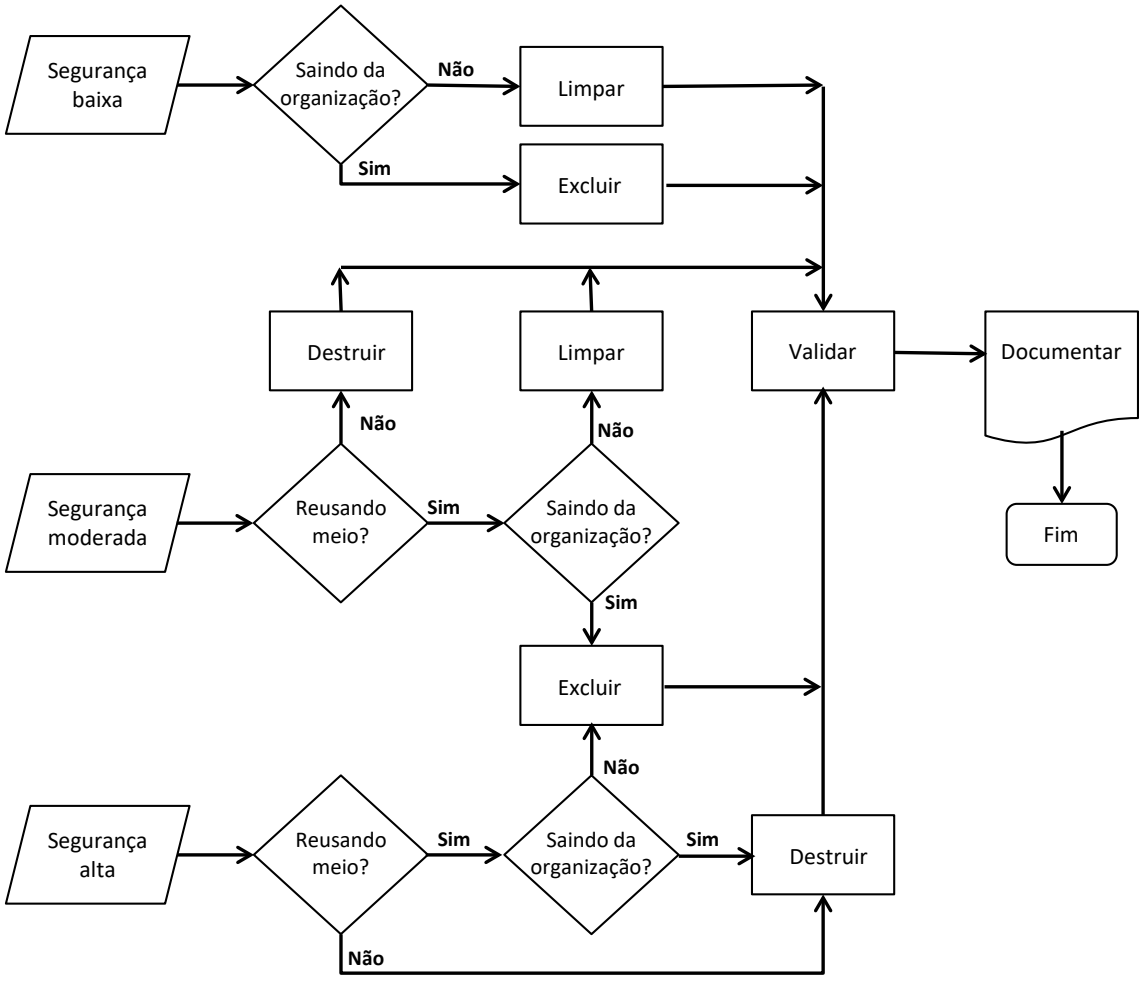


Figura 7 – Fluxo de sanitização e destinação de mídias [40]

As recomendações NIST para limpeza de meios magnéticos sugerem uma passagem de sobrescrita com um padrão fixo, como tudo zeros. Múltiplas passagens de sobrescrita são opcionais. Para limpeza de smartphones sugere-se reiniciar o dispositivo para o padrão de fábrica (*reset to*

factory). Não há como realizar limpeza sobre meios ópticos, tais como CD, DVD e Bluray. Para esses meios, as recomendações somente preveem destruição.

Dispositivos de armazenamento que empregam memória *flash* tem tratamento diferenciado, de acordo com o tipo do dispositivo. Para discos SSD, por exemplo, as seguintes ações são sugeridas:

- Limpeza: no mínimo uma passagem de escrita com um valor fixo de dado, como tudo zeros. Passagens múltiplas ou valores mais complexos podem ser utilizados. É importante notar que sobrescrita em mídia baseada em *flash* reduz significativamente o tempo de vida desta mídia. Ao invés de uma sobrescrita, pode ser usado o comando *SECURITY ERASE UNIT*, se ele for suportado pelo disco.
- Exclusão: utilizar o comando de apagar todas as Unidades de Apagamento, ou , se o disco suporta criptografia, utilizar apagamento criptográfico. Opcionalmente, após o apagamento pode-se escrever um padrão no disco e realizar um segundo apagamento.
- Destruição: triturar, pulverizar, desintegrar ou incinerar o disco.

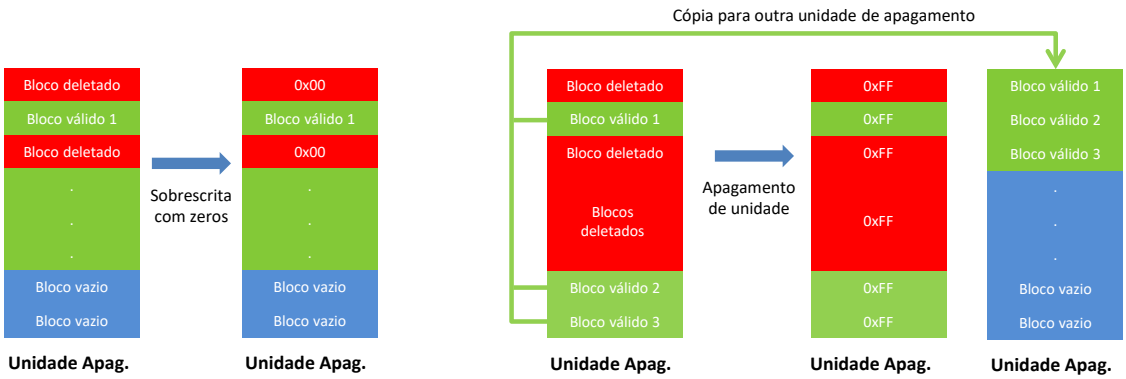
Para dispositivos *flash* removíveis, tal como *pen drives*, *drives* de memória *flash*, *memory sticks*, as seguintes ações são sugeridas:

- Limpeza: sobrescrever a mídia com um valor de dados fixo e aprovado pela organização. O padrão de limpeza deve ser, pelo menos, duas passagens, para incluir um padrão na primeira passagem e o seu complemento na segunda passagem. Podem ser usadas passagens adicionais.
- Exclusão: a maioria das mídias removíveis USB não suporta comandos de limpeza, ou se suportado, as interfaces não são suportadas de forma padronizada em todos estes dispositivos.
- Destruição: triturar, pulverizar, desintegrar ou incinerar o dispositivo.

Para uma relação completa de todos os dispositivos tratados, recomenda-se a leitura das Normas para Sanitização de Mídia [40].

4.2 **Método de apagamento híbrido (Método de Sun)**

Sun [51] propôs um esquema híbrido adaptativo que combina *zero-overwriting* e *erase blocks*. *Zero-overwrite* é um método de eliminação que substitui blocos com 0x00 para que os dados existentes possam ser apagados de forma segura. Em contraste, *erase blocks* exclui todos os dados que são armazenados em uma unidade de apagamento através da realização de uma operação de apagamento. Neste momento, se *blocos* válidos permanecem nas unidades de apagamento, estes *blocos* devem ser transferidos para outra unidade de apagamento, são necessárias operações adicionais de modo de leitura / gravação. Como pode ser visto na Figura 8 a seguir:



(a) Sobrescrita (b) Apagamento

Figura 8 – Processo do método Híbrido

O esquema híbrido adaptativo proposto por Sun [51] combina os dois métodos anteriores para minimizar o custo da operação, realizando os passos abaixo.

1. Procura um *bloco* a ser deletado.
2. Verifica se o custo da substituição por zeros (*zero-overwriting*) nos blocos deletados é mais barato do que apagar a unidade de apagamento (*erase blocks*) que contém os blocos excluídos. Se o custo de substituir é mais barato do que apagar a unidade de apagamento, a substituição com zeros é executada.
3. Caso contrário, os *blocos* válidos são copiados para outra unidade de apagamento e o apagamento da unidade é aplicado para excluir dados.

A operação de apagamento converte todos os bits de '0' a '1' em uma unidade de apagamento, enquanto a operação de gravação muda seletivamente bits do '1' para '0' em um *bloco.* Devido a este mecanismo, no *erase blocks*, a unidade de apagamento no qual a operação de apagamento foi realizada é preenchido com 0xFF. Além disso, *blocos* válidos que estão contidos na unidade de apagamento excluídos são movidos para outra unidade de apagamento para preservar os dados armazenados nos blocos válidos.

O método de eliminação, de Sun [51], é selecionado de acordo com o número de blocos eliminados. Em outras palavras, se o custo de substituição de zero é menor do que a da unidade de apagamento, *zero-overwriting* é realizada. Por outro lado, se há mais *blocos* excluídos do que *blocos* válidos em uma unidade de apagamento, a substituição é mais eficiente.

4.3 **Método de criptografia (método de J. Lee)**

Em contraste ao método de Sun [51], o método utilizado por Lee [29] [30] propõe utilizar criptografia para realizar uma remoção segura de dados.

As chaves de criptografia são geradas aleatoriamente, e os dados são criptografados usando essas chaves de criptografia. As chaves de criptografia de um arquivo específico são armazenadas em uma unidade de apagamento (EU), como pode ser visto na Figura 9.

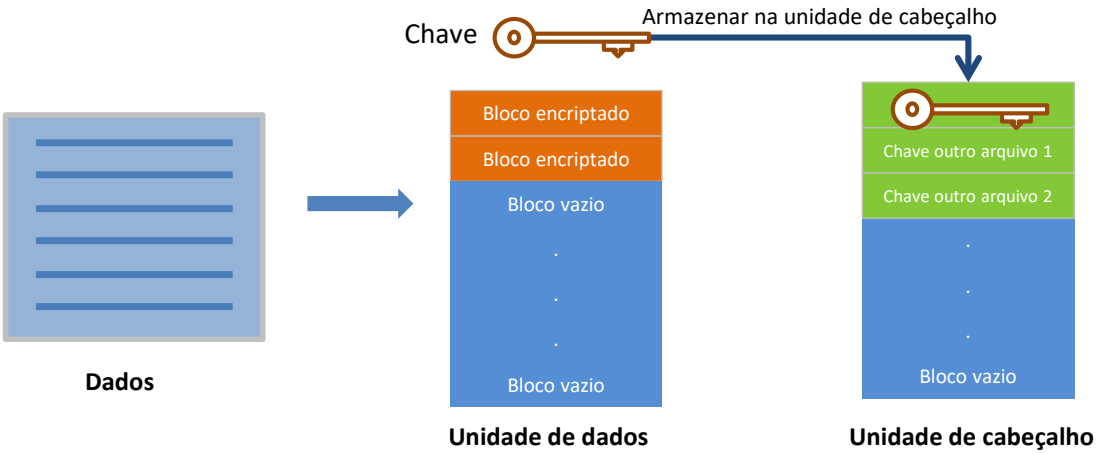


Figura 9 – Processo de armazenamento de dados com criptografia.

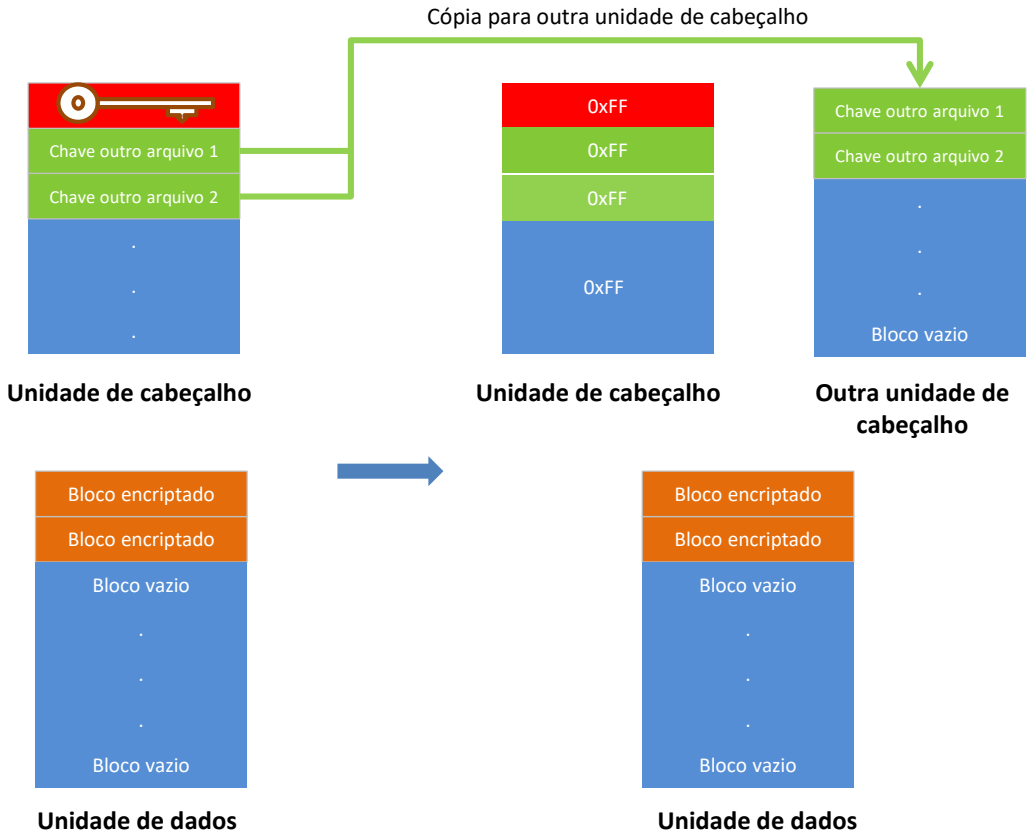


Figura 10 – Processo de remoção de dados no método cifrado

Portanto, somente uma operação de apagamento é necessária para apagar dados de forma segura. A Figura 10 mostra que, durante o processo de exclusão, a chave de um arquivo específico é apagada. Embora os dados do arquivo excluído ainda permaneçam na memória *flash*, os dados do arquivo são criptografados, e as chaves de decodificação foram eliminadas, de modo que se pode considerar o arquivo excluído como um arquivo seguramente deletado. O mecanismo de remoção do método de Lee funciona da seguinte forma:

1. Pesquisar uma unidade de cabeçalho que armazena as chaves de criptografia de dados apagados.

2. Verificar se esta unidade de apagamento contém outras chaves de criptografia válidas, além das chaves que devem ser excluídas. Se chaves de criptografia válidas estão armazenadas, essas chaves são copiadas para outra unidade de apagamento.
3. Apagar a unidade de apagamento o qual foi copiado no passo anterior.

Além disso, a unidade de apagamento que armazena chave de criptografia de um arquivo específico também tem meta-dados desse arquivo, portanto, após a eliminação, tanto as chaves de criptografia de arquivos e todos os meta-dados associados a este arquivo são seguramente removidos.

4.4 **Apagamento duplo (método de Huang)**

Huang [17] propõe um método que utiliza duas passagens. Na primeira passagem, os blocos do arquivo são sobrescritos com zero (*zero overwriting*). Na segunda passagem, realiza-se o apagamento da unidade de apagamento que contém estes blocos, após copiar os blocos válidos (de outros arquivos) para uma nova posição. Huang realiza estas duas passagens para ficar em conformidade com os principais padrões sugeridos para limpeza de dados, mostrados na Tabela 4.

Tabela 4 – Principais Padrões de Limpeza de dados

Padrão de Limpeza	Descrição
DoD 5220.22-M-Sup 1 (1995) [39]	1. Apagar todos os dados; 2. Sobrescrever com um único caractere; 3. Sobrescrever com o complemento daquele valor; 4. Sobrescrever com um único caractere randômico; 5. Verificar se procedimentos adicionais são necessários.
DoD 5220.22-M (1997)	1. Sobrescrever com um único caractere; 2. Apagar a área sobrescrita.
NSA/CSS storage device declassification manual [41]	1. Sobrescrever com um padrão pré-determinado; 2. Verificar a área sobrescrita com leituras randômicas.
Media Clearing, Purging and Destruction [8]	1. Apagar todos os dados; 2. Sobrescrever duas vezes com valores pseudorrandômicos; 3. Sobrescrever com um padrão conhecido; 4. Verificar se procedimentos adicionais são necessários.
IRIG 106-07, CH.10 [21]	1. Apagar todos os dados; 2. Sobrescrever com 0x55; 3. Apagar todos os dados novamente; 4. Sobrescrever com 0xAA, e pagar uma terceira vez; 5. Sobrescrever com uma sequência randômica.

Este método utiliza basicamente as mesmas operações que o método de Sun (Seção 4.3). Entretanto, sempre realiza uma sobrescrita com zeros e após um apagamento, ao invés de analisar qual das duas operações tem menor custo e realizar somente esta operação.

Se por um lado o método sugerido por Huang procura seguir os padrões internacionais para oferecer mais segurança na remoção de um arquivo, por outro lado este método sacrifica a vida útil de uma memória *flash* ao realizar mais operações de escrita e apagamento.

4.5 **Método de criptografia com apagamento (método de B. Lee)**

O método de B. Lee et all [28] utiliza apagamento criptográfico e procura ficar em conformidade com os mesmos padrões usados pelo método de Huang (ver Seção 4.5 e Tabela 4). Para maior segurança na remoção da chave criptográfica, o método de B. Lee realiza uma sobrescrita da chave com zeros (Figura 11) e a seguir efetua um apagamento da chave (Figura 12).

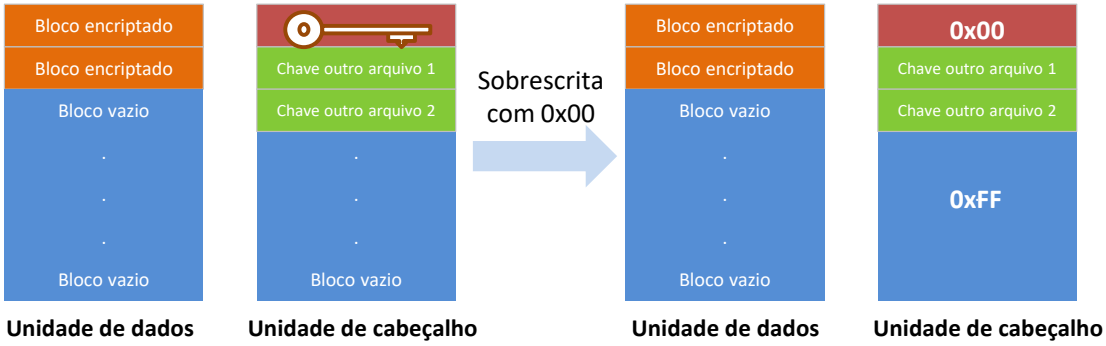


Figura 11 – Processo de sobrescrita da chave no método de B. Lee

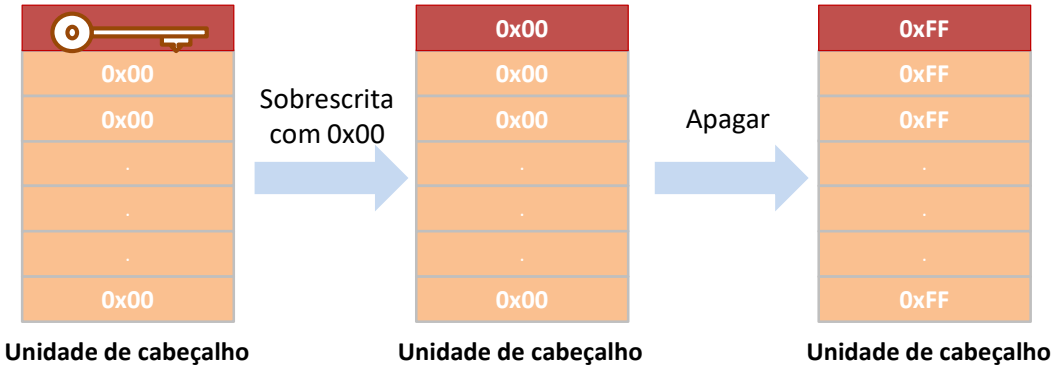


Figura 12 – Processo de apagamento no método de B. Lee

Mais detalhadamente, o método proposto segue os seguintes passos:

1. Procura por uma unidade de cabeçalho que contenha a chave criptográfica do arquivo a ser removido.
2. Sobrescreve esta chave com um padrão de 0x00 (Figura 11).
3. Verifica se existem outras chaves ainda válidas na unidade de cabeçalho. Se existirem, o processo de remoção termina.

4. Se não existem mais chaves válidas, então se realiza um apagamento da unidade de cabeçalho (Figura 12).

Observe-se que o método proposto per B. Lee pretende seguir as normas de limpeza e realizar duas operações (sobrescrita e apagamento), mas a operação de apagamento somente é realizada quando todas as chaves da EU de cabeçalho não forem mais válidas. Com isto a realização da segunda operação (apagamento) é retardada até uma eventual remoção de outros arquivos invalide as demais chaves da EU. Como o tempo para que isto ocorra é indeterminado, o método proposto na realidade não segue estritamente as normas de remoção (Tabela 4).

4.6 **Análise comparativa**

Os três métodos descritos nas seções anteriores abrangem as técnicas atualmente utilizadas para obter uma remoção segura de arquivos. Existem variações destes métodos, como por exemplo o método de Joel [23], que também opta por utilizar criptografia e controle de chaves para deixar os blocos "apagados" devidamente seguros, mas todos usam as três técnicas básicas descritas. Uma análise comparativa resumida destas três técnicas pode ser vista na Tabela 5.

Tabela 5 – Análise comparativa das técnicas de remoção segura de arquivos

	Sobrescrita	Apagamento	Criptografia
Aplicável a meio magnético?	Sim	Sim, mas com baixo desempenho	Sim
Aplicável a memória *flash*?	Não	Sim	Sim
Acelera o desgaste do meio?	Sim, porque usa várias escritas	Sim, mas com menor impacto porque usa unidades de apagamento	Não
Necessita estrutura extra de gerenciamento?	Não	Sim, para os blocos a serem removidos	Sim, para as chaves de criptografia
Impacta no desempenho do sistema de arquivos	Não	Não	Sim

Como pode ser observado na Tabela 5, os métodos tradicionais de remoção de arquivos através de sobrescrita não se aplicam a memórias *flash* e, portanto, é necessário investigar mais a fundo o problema da remoção segura de arquivos em memórias *flash*. A ideia de realizar uma operação de sobrescrita para eliminar dados anteriores é atraente e inclusive recomendada pelas normas (ver Tabela 4), mas deve-se considerar que a camada FTL de uma *flash* redireciona a operação de sobrescrita de um mesmo bloco lógico para um bloco físico distinto. Assim, a sobrescrita deve ser feita não no nível de blocos lógicos, mas sim sobre blocos físicos.

Em relação aos padrões recomendados para limpeza de dados, a Tabela 6 apresenta uma comparação dos métodos descritos nas seções anteriores.

Tabela 6 – Comparação entre recomendações e métodos de remoção

	Apagar todos os dados	Sobrescrita com valor conhecido	Sobrescrita com complemento do valor	Sobrescrita com valor randômico
Recomendação DoD 1995	Sim	Sim	Sim	Sim
Recomendação DoD 1997	Sim	Sim	Não	Não
Recomendação NSA/CSS	Não	Sim	Não	Não
Recomendação Media Clearing	Sim	Sim	Não	Sim
Método de Sun, sobrescrita	Não	Sim	Não	Não
Método de Sun, apagamento	Sim	Não	Não	Não
Método de J. Lee	Sim	Não	Não	Não
Método de Huang	Sim	Sim	Não	Não
Método de B.Lee	Sim, mas não imediatamente	Sim	Não	Não

Observe-se que os padrões sugeridos pelo NIST (Seção 4.2) sugerem no mínimo duas passagens, com valores complementares. Como em memória *flash* uma sobrescrita usa o padrão 0x00 e um apagamento restaura a memória para o valor 0xFF, para atender os padrões NIST é necessário realizar uma sobrescrita seguida de um apagamento. Neste sentido, os únicos métodos que atendem os padrões NIST são o método de Huang e o Método de B. Lee (mas o método de B. Lee posterga o apagamento para um momento indeterminado).

4.7 **Método Proposto**

O método híbrido de remoção de arquivos proposto por Sun utiliza uma função de custo para decidir entre a realização de uma sobrescrita ou de um apagamento. Os custos são basicamente calculados a partir do tempo necessário para realização das operações, e o método utiliza um fator de benefício para o apagamento, para considerar o fato de blocos apagados estarem prontos para serem reutilizados, enquanto o mesmo não corre com blocos sobrescritos.

Entretanto, como será analisado na Seção 5.4.8 do capítulo seguinte, o método de Sun não considera a existência de setores livres, ainda não utilizados para uma escrita, quando calcula o custo de um apagamento. Da mesma forma, também não considera que blocos removidos por sobrescrita deverão ser futuramente reciclados por um apagamento antes de estarem novamente disponíveis para uso. Assim, este trabalho propõe uma análise mais detalhada das funções de custo.

Para o cálculo dos custos, propõe-se utilizar os seguintes fatores:

- ZO: tempo necessário para sobrescrever os blocos a serem removidos.
- A: tempo necessário para apagar uma unidade que contenha blocos a serem removidos, acrescido do tempo necessário para copiar os blocos válidos para outra unidade de apagamento.
- B: fator de benefício ao apagar uma unidade, pois após um apagamento todos os blocos desta unidade estão disponíveis para uso.
- P: fator de penalidade. Em um apagamento, esta penalidade reflete a existência de blocos livres em uma unidade, que serão apagados prematuramente. Em uma sobrescrita, esta penalidade reflete o fato de blocos sobrescritos necessitarem de uma reciclagem (apagamento) futura.

Considerando as combinações possíveis com estes fatores, os seguintes métodos híbridos podem ser considerados:

- ZO-AB: método que compara custos de sobrescrita com zeros (ZO) e de apagamento (A), mas considera um Benefício (B) no custo ao realizar apagamentos (método originalmente proposto por Sun)
- ZO-A: método sem considerar Benefício no apagamento.
- ZOP-ABP: método considerando um custo de Penalidade tanto para sobrescrita com zeros como para apagamento.
- ZO-ABP: método com Penalidade somente para apagamento.
- ZOP-AB: método com Penalidade para sobrescrita e apagamento, mas sem considerar o Benefício no apagamento.
- ZO-AP: método de Sun Penalidade somente para apagamento, e sem considerar Benefício para este apagamento.
- ZOP-AB: método com Benefício para apagamento e Penalidade para sobrescrita com zeros.
- ZOP-A: método sem Benefício para Apagamento e com Penalidade para sobrescrita com zeros.

Para escolher o método mais adequado, realiza-se no Capítulo 5 a seguir uma análise dos métodos discutidos neste capítulo, utilizando uma modelagem analítica. Uma análise quantitativa é realizada no Capítulo 7, utilizando-se o Simulador desenvolvido no Capítulo 6.

5 ANÁLISE DOS MÉTODOS DE REMOÇÃO

O ponto principal deste trabalho trata da eliminação segura de arquivos. Como um sistema operacional faz a remoção de um arquivo simplesmente liberando os blocos que o arquivo ocupava em disco (*deletion by unlinking*), a sua recuperação com ferramentas forenses é possível, desde que os blocos ainda não tenham sido alocados e sobrescritos por outro arquivo. Para a eliminação segura, ou seja, para impedir que um arquivo seja recuperado por técnicas forenses, três métodos básicos que podem ser utilizados:

1. Uso de um sistema de arquivos seguro, que proteja todos os blocos do sistema de arquivos, tanto aqueles que estejam em uso como aqueles que foram liberados porque um arquivo foi removido.
2. Uso de criptografia para realizar a eliminação, onde um arquivo é armazenado de forma criptografada e a sua remoção é realizada eliminando-se a chave de criptografia.
3. Uso de sobrescrita para realizar a eliminação, onde os blocos pertencentes a um arquivo que deve ser removido são sobrescritos com padrões binários.

Como não é objetivo do presente trabalho implementar ou analisar sistemas de arquivos seguros, mas sim realizar a remoção segura de arquivos, limitou-se a pesquisa realizada nos métodos que realizam eliminação criptográfica ou eliminação por sobrescrita. Nas seções seguintes estes métodos são analisados em maior detalhe.

5.1 **Uso de sistema de arquivos seguro**

Um sistema de arquivo seguro, que use métodos criptográficos para cifrar e decifrar dados quando acessa blocos de um arquivo impede o acesso não autorizado aos arquivos. Mesmo que o sistema operacional realize a remoção de um arquivo simplesmente liberando os blocos em disco (*deletion by unlinking*), a recuperação destes blocos é dificultada pelo fato dos blocos estarem cifrados, e sem a chave de criptografia correspondente ao conteúdo original dos dados não pode ser recuperado.

Assim, a utilização de um sistema de arquivos seguro tem como vantagem o fato de proteger por criptografia tanto os dados existentes como os dados removidos. Além disto, sistemas de arquivos seguros são uma metodologia tradicional, bastante usada e difundida, com vários sistemas disponíveis.

Entretanto, vários outros fatores devem ser considerados, que podem representar uma desvantagem. O uso de métodos criptográficos afeta negativamente o desempenho, pois cada operação de leitura e escrita de blocos precisa de tempo adicional para realizar a cifragem dos dados e sua respectiva decifragem. Para acessar os arquivos, o usuário deve gerenciar chaves de criptografia, ou então gerenciar credenciais de autenticação. A perda destas credenciais impede o acesso aos arquivos. Além disto, um erro que afete um bloco impede a decifragem correta deste bloco. Neste caso, a segurança obtida afeta negativamente a tolerância contra falhas [50] [61].

5.2 **Eliminação por criptografia**

Diversos métodos propostos, entre eles [23], [29], [44] e [15], usam criptografia para obter uma eliminação segura. Blocos distintos são escritos em disco cifrados por chaves distintas, e quando um bloco deve ser eliminado simplesmente se elimina a chave. Nestes métodos, o problema de eliminação segura de um arquivo é substituído pelo problema de eliminação segura de uma chave criptográfica. Assim, não é mais necessário eliminar grandes quantidades de dados, mas sim eliminar uma chave de 128 a 256 bits [15]. Outros tamanhos de chaves podem ser utilizados, como por exemplo 512 bits, mas o problema permanece basicamente o mesmo, ou seja, eliminar uma chave de forma segura.

Estes métodos têm em comum a vantagem de utilizar algoritmos de criptografia para cifrar blocos antes de escrevê-los na memória *flash* e, como em um sistema de arquivos seguro, tanto dados existentes como dados a serem removidos estão protegidos por criptografia. Se o único objetivo é realizar uma eliminação segura, as chaves são geradas automaticamente através de números randômicos pelo sistema de proteção, e gerenciadas pelo próprio sistema, de forma que o usuário não precisa gerenciar chaves. Neste caso, o uso de criptografia é feito de forma transparente para o usuário.

Como desvantagens, existe a queda de desempenho no acesso aos arquivos, tal como em um sistema de arquivos seguro. Além disto, as chaves devem ser armazenadas em algum lugar da memória *flash*, e precisam ser eliminadas de forma segura. A simples sobrescrita de uma chave não é efetiva em uma memória *flash*, pois para evitar desgaste prematuro da mídia escritas em um mesmo bloco lógico são mapeadas para escritas em blocos físicos distintos [9]. Soluções como a proposta por [29] procuram resolver este problema, mas necessitam de uma análise mais detalhada para verificar sua eficácia em termos de segurança e tolerância a falhas. Esta análise será conduzida durante a realização do trabalho proposto, como descrito nos objetivos da Seção 1.2.

5.3 **Eliminação por sobrescrita**

Técnicas normais de sobrescrita em meios magnéticos não são aplicáveis em memória *flash*, devido ao mecanismo para impedir o desgaste prematuro da mídia [9]. Como a memória *flash* não pode executar atualizações de blocos (ou seja, *overwrite*), antes de escrever um novo bloco é necessário que seja realizada uma operação de apagamento de blocos, ou seja, apagar toda a unidade de apagamento que contém estes blocos. Mas unidades de apagamento têm uma granularidade maior do que os blocos de gravação, e com isto uma única unidade de apagamento pode armazenar dados para arquivos diferentes, e por isso só pode ser apagada quando todos os seus dados são marcados como excluídos ou quando estes dados foram replicados em outro lugar. Além disso, a memória *flash* degrada a cada apagamento; apagamentos frequentes encurtam a vida útil do dispositivo. Portanto, a solução simples de apagar qualquer unidade de apagamento que contém dados que foram excluídos é muito cara em relação ao tempo e ao desgaste do dispositivo [51]. Assim, devem ser desenvolvidos métodos como o de [51], que procuram realizar uma gerência eficiente dos blocos da memória *flash*, de forma a reduzir o número de apagamentos necessários ao mesmo tempo em que realizam a eliminação segura por sobrescrita

com zeros. Pelas características tecnológicas de uma memória *flash*, um bloco não pode ser escrito antes de ter sido apagado, mas o seu conteúdo pode ser modificado alterando todos os bits em “1” para “0”. Isto não elimina a necessidade de apagamento, mas permite a eliminação do conteúdo de um bloco.

Métodos como o de [51] tem como vantagem o fato de não necessitar de métodos criptográficos nem precisar gerenciar chaves de cifragem. Não há tempo adicional de cifragem de dados, pois os dados são lidos normalmente.

Entretanto, estes métodos necessitam de algoritmos de gerenciamento de blocos e de unidades de apagamento, pois ao mesmo tempo em que devem garantir que dados removidos não possam ser recuperados, também não devem utilizar muitas operações de apagamento, pois isto reduz a vida útil da memória *flash*. Por intercalarem operações de leitura e escrita com operações de apagamento, que são mais lentas, estes métodos podem impactar negativamente no desempenho global de acesso aos arquivos. Também podem subutilizar o espaço da memória, ao alocar arquivos em *erase blocks*, e não mais em blocos normais. Uma análise mais detalhada destes métodos, com especial atenção quanto a sua eficiência em termos de gerência de arquivos será realizada durante o decorrer do trabalho.

5.4 **Análise**

Independente do sistema de arquivos utilizado pelo sistema operacional de uma máquina, um arquivo pode ser modelado como uma lista de blocos, onde um bloco é uma unidade de alocação, ou seja, a menor porção do meio de armazenamento que pode ser reservada para guardar um arquivo.

Em um disco magnético, os blocos gerenciados pelo sistema de arquivo correspondem diretamente aos blocos físicos no meio magnético. Em uma memória *flash*, entretanto, a FTL mapeia blocos do sistema de arquivos (blocos virtuais) para blocos físicos na mídia, conforme visto na Seção 3.1.

Ao remover arquivo, o sistema operacional não remove os blocos do arquivo do meio de armazenamento, e portanto o conteúdo destes blocos pode ser recuperado por ferramentas forenses enquanto não forem reutilizados para outro arquivo. Para impedir esta recuperação, os métodos analisados no Capítulo 4 usam operações de sobrescrita e de criptografia. Estas operações introduzem um tempo extra de processamento e, no caso específico de uma memória *flash*, a sobrescrita pode ser realizada através de *zero-overwriting* ou de apagamento (*erase blocks*). Para analisar o impacto deste processamento adicional e dos métodos de sobrescrita em *flash*, é necessária uma análise das operações básicas envolvidas.

As grandezas utilizadas são as seguintes:

- L_t : Tempo de uma leitura (valor de referência básica).
- E_t : Tempo de uma escrita (em uma memória *flash*, varia de 8 a 11 vezes o tempo de uma leitura).

- A_t : Tempo de um apagamento (em uma memória *flash*, varia de 16 a 100 vezes o tempo de uma leitura).
- C_t : Tempo de cifragem (o quanto demora cifrar um bloco de leitura/escrita – tempos para o AES e um bloco de 4KB).
- D_t : Tempo de decifragem (o quanto demora decifrar um bloco de leitura/escrita – tempos para o AES e um bloco de 4KB).
- $L_{arquivo}$: Tempo de leitura de um arquivo.
- $E_{arquivo}$: Tempo de escrita de um arquivo.
- $R_{arquivo}$: Tempo de remoção de um arquivo.
- $N_{blocos\_arquivo}$: Número de blocos lógicos do arquivo a ser removido. Corresponde ao número de blocos físicos a serem removidos da *flash*.
- $N_{blocos\_unidade}$: Número de blocos de leitura/escrita por unidade de apagamento da *flash*. É uma constante para toda a mídia, definida pela FTL.
- $N_{blocos\_a\_remover}$: Número de blocos de leitura/escrita a serem removido em uma determinada unidade de apagamento.
- $N_{blocos\_válidos}$: Número de blocos de leitura/escrita que ainda são válidos dentro de uma determinada unidade de apagamento.
- N_{blocos_livres} : Número de blocos de leitura/escrita marcados como livres (*free*) em uma determinada unidade de apagamento.

Observe-se que o tempo de remoção do arquivo pelo sistema operacional depende do sistema de arquivos utilizado, mas é igual para todos os métodos, e portanto não será considerado neste trabalho.

Como uma memória *flash* é organizada em unidades de apagamento (ver Figura 4), os números de blocos descritos acima devem ser definidos para cada unidade de apagamento. Neste trabalho é utilizado um índice quando for necessário referenciar uma unidade de apagamento específica, como por exemplo:

- $N_{blocos\_livres}(i)$: Número de blocos livres na unidade de apagamento de índice i.

Diversos cenários distintos são analisados nas seções a seguir.

5.4.1 Remoção normal

Como já foi analisado, uma remoção pelo sistema operacional não é segura, pois os blocos do arquivo removido podem ser recuperados por técnicas forenses. Por outro lado, esta remoção é rápida e eficiente, pois não requer nenhuma operação extra além da remoção do descritor do arquivo no diretório (*unlinking*) e a liberação dos blocos do arquivo, marcando-os como livres (*free*).

Tempos:

- $L_{arquivo} : N_{blocos\_arquivo} * L_t$
- $E_{arquivo} : N_{blocos\_arquivo} * E_t$

- $R_{arquivo}$: 0 (somente o tempo de remoção pelo sistema operacional)

Ameaça: Blocos do arquivo não são apagados, podem ser recuperados por ferramentas forenses enquanto outro arquivo não for criado nos mesmos blocos.

5.4.2 Remoção em um sistema de arquivos seguro

Um sistema de arquivos seguro utiliza criptografia, e portanto deve-se adicionar o custo de cifrar e decifrar blocos, assim como o custo de gerenciar as chaves criptográficas utilizadas.

Tempos:

- $L_{arquivo}$: $N_{blocos\_arquivo} * (L_t + D_t)$
- $E_{arquivo}$: $N_{blocos\_arquivo} * (E_t + C_t)$
- $R_{arquivo}$: 0 (somente o tempo de remoção pelo sistema operacional)

Ameaças:

Dependendo de como as chaves de criptografia são gerenciadas (se a chave é baseada nas credenciais do usuário, como nome e senha), um arquivo deletado pode ser recuperado por alguém que invada a conta do usuário. Além disto, deve ser considerado o fato de um algoritmo ser quebrado no futuro ou de uma chave ser descoberta no futuro.

5.4.3 Remoção por sobrescrita

Pelas características da FTL, remover blocos por sobrescrita não é uma técnica efetiva, uma vez que o mapeamento da FTL redirecionaria a operação de sobrescrita para outro bloco lógico, e não para o bloco físico que se deseja sobrescrever. Entretanto, realiza-se uma análise para este caso, pois o seu custo fornece um valor teórico ideal para métodos de sobrescrita.

Assim, o tempo de sobrescrita é dado pelo número de blocos do arquivo e pelo tempo de realização de uma escrita:

$$T_{sobrescrita} = N_{blocos\_arquivo} * E_t$$

Tempos:

- $L_{arquivo}$: $N_{blocos\_arquivo} * L_t$
- $E_{arquivo}$: $N_{blocos\_arquivo} * E_t$
- $R_{arquivo}$: $T_{sobrescrita}$

Ameaça:

Enquanto todos os blocos não forem sobrescritos, partes do arquivo podem ser recuperadas. Mas o tempo de sobrescrita é significativamente curto, e depende basicamente do tamanho do arquivo.

5.4.4 Remoção pelo método de Sun

O método de Sun [51] (Seção 4.3) analisa para cada unidade de apagamento qual o tempo necessário para uma sobrescrita e qual o tempo para um apagamento.

O tempo de uma sobrescrita é o tempo necessário para realizar uma escrita com *zero-overwriting*:

$$T_{zo}(i) = N_{blocos\_a\_remover}(i) * E_t$$

$$T_{zo}total = \Sigma\ T_{zo}(i)$$

O tempo de apagamento é o tempo necessário para copiar os blocos ainda válidos para outra unidade de apagamento e a seguir apagar a unidade:

$$T_A(i) = N_{blocos\_válidos}(i) * (L_t + E_t) + A_t$$

$$T_A total = \Sigma\ T_A(i)$$

O método considera ainda um benefício associado à operação de apagamento, pois cria blocos livres que podem ser usados futuramente. Já uma operação de sobrescrita não cria nenhum bloco livre, então seu benefício futuro é zero:

$$Benefício_{zo}(i) = 0$$

$$Benefício_A(i) = (N_{blocos\_ganhos}(i) / N_{blocos\_unidade}) * A_t$$

onde $N_{blocos\_ganhos}(i) = N_{blocos\_unidade} - N_{blocos\_válidos}(i)$

então $Benefício_A(i) = ((N_{blocos\_unidade} - N_{blocos\_válidos}(i)) / N_{blocos\_unidade}) * A_t$

ou $Benefício_A = (1 - (N_{blocos\_válidos}(i) / N_{blocos\_unidade})) * A_t$

Portanto, o benefício de um apagamento vai aumentando a medida que o número de blocos válidos na unidade de apagamento vai diminuindo.

Se $N_{blocos\_válidos}(i) = N_{blocos\_unidade}$, então:

$Benefício_A = (1-(N_{blocos\_válidos}(i)/N_{blocos\_unidade})*A_t = (1-(N_{blocos\_unidade}/N_{blocos\_unidade})*A_t = (1-1)*A_t = 0$

Por outro lado, se $N_{blocos\_válidos}(i) = 0$, então:

$Benefício_A = (1 - (N_{blocos\_válidos}(i) / N_{blocos\_unidade}) * A_t = (1 - (0 / N_{blocos\_unidade}) * A_t = (1-0)*A_t = A_t$

Ou seja, o benefício de um apagamento ($Benefício_A$) varia entre zero e A_t. Já o benefício de uma sobrescrita é sempre zero, como já foi dito anteriormente.

Assim, o custo de uma operação ou de outra é definido como o tempo necessário para realizar a operação, subtraído do benefício obtido. Para a sobrescrita, tem-se:

$$Custo_{zo}(i) = T_{zo}(i) - Benefício_{zo}(i)$$

$$Custo_{zo}(i) = T_{zo}(i)$$

$$Custo_{zo}(i) = N_{blocos\_a\_remover}(i) * E_t$$

E para o apagamento tem-se:

$$Custo_A(i) = T_A(i) – Benefício_A(i)$$

$$Custo_A(i) = (N_{blocos\_válidos}(i) * (L_t + E_t) + A_t) – ((1 – (N_{blocos\_válidos}(i) / N_{blocos\_unidade}) * A_t)$$

$$Custo_A(i) = N_{blocos\_válidos}(i) * (L_t + E_t) + A_t – (A_t – (N_{blocos\_válidos}(i) / N_{blocos\_unidade}) * A_t)$$

$$Custo_A(i) = N_{blocos\_válidos}(i) * (L_t + E_t) + A_t – A_t + (N_{blocos\_válidos}(i) / N_{blocos\_unidade}) * A_t$$

$$Custo_A(i) = N_{blocos\_válidos}(i) * (L_t + E_t) + (N_{blocos\_válidos}(i) / N_{blocos\_unidade}) * A_t$$

$$Custo_A(i) = N_{blocos\_válidos}(i) * ((L_t + E_t) + A_t / N_{blocos\_unidade})$$

Um uso simplificado do método de Sun, seria somar os custos de sobrescrita e de apagamento de cada unidade de apagamento, e a seguir realizar para todas as unidades sempre sobrescrita ou sempre apagamento, baseado na operação de menor custo global. O método híbrido proposto, entretanto, realiza a comparação dos dois para cada unidade de apagamento, e usa nesta unidade a operação de menor custo.

Entretanto, a análise realizada por este método não leva em conta que uma unidade de apagamento também pode ter blocos livres, e não somente blocos a serem removidos e blocos válidos. Isto será tratado na Seção 5.4.8.

Tempos:

- $L_{arquivo} : N_{blocos\_arquivo} * L_t$
- $E_{arquivo} : N_{blocos\_arquivo} * E_t$
- $R_{arquivo} : \Sigma \min (Custo_{zo}(i), Custo_A(i))$

Ameaça:

Assim como no método teórico de sobrescrita, enquanto todos os blocos não forem apagados, partes do arquivo podem ser recuperadas. E o apagamento deve ser realizado por todos os blocos do arquivo.

5.4.5 Remoção pelo método de J. Lee

O método de J. Lee [29] (Seção 4.4) utiliza apagamento criptográfico, e para remover a chave de criptografia, primeiro copia as demais chaves válidas da EU de cabeçalho, e a seguir realiza uma operação de apagamento nesta unidade. Observe-se que só é necessário o apagamento de uma unidade (aquela que contém a chave a ser removida). Assim:

$$T_A = N_{chaves\_válidas} * (L_t + E_t) + A_t$$

Tempos:

- $L_{arquivo} : N_{blocos\_arquivo} * (L_t + D_t)$

- $E_{arquivo}$: $N_{blocos\_arquivo}$ * (E_t + C_t)
- $R_{arquivo}$: T_A = $N_{chaves\_válidas}$ * (L_t +E_t) + A_t

Ameaça:

Como o método envolve somente o apagamento de uma unidade de apagamento, ele é extremamente rápido. Entretanto, os blocos do arquivo não são removidos, e, pelo menos em termos teóricos, podem vir a ser recuperados em um tempo futuro.

5.4.6 Remoção pelo método de Huang

O método duplo de Huang [17] (Seção 4.5) realiza tanto uma operação de sobrescrita como uma operação de apagamento, e portanto o seu tempo é dado pelo somatório da realização destas das operações em todas unidades de apagamento que contêm blocos do arquivo a ser removido. Assim, tem-se para a sobrescrita:

$$T_{zo}(i) = N_{blocos\_a\_remover}(i) * E_t$$

$$T_{zo}total = \Sigma\ T_{zo}(i)$$

E para o apagamento:

$$T_A(i) = N_{blocos\_válidos}(i) * (L_t + E_t) + A_t$$

$$T_A total = \Sigma\ T_A(i)$$

O método proposto por Huang não leva em conta o benefício obtido por apagar blocos, e também não considera o fato de uma unidade de apagamento poder conter blocos livres. Isto será tratado na Seção 5.4.8.

Tempos:

- $L_{arquivo}$: $N_{blocos\_arquivo}$ * L_t
- $E_{arquivo}$: $N_{blocos\_arquivo}$ * E_t
- $R_{arquivo}$: $T_{zo}total + T_A total$

Ameaça:

Assim como no método teórico de sobrescrita, enquanto todos os blocos não forem apagados, partes do arquivo podem ser recuperadas. E o apagamento deve ser realizado por todos os blocos do arquivo.

5.4.7 Remoção pelo método de B. Lee

O método proposto por B. Lee [28] (Seção 4.6) utiliza apagamento criptográfico e realiza inicialmente uma sobrescrita da chave e, quando todas as chaves da mesma EU de cabeçalho já estiverem eliminadas, executa uma operação de apagamento nesta EU. Assim, o tempo do método é inicialmente o tempo de uma sobrescrita. Posteriormente, quando o apagamento é realizado, deve-se somar o tempo deste apagamento.

Tempos:

- $L_{arquivo}$: $N_{blocos\_arquivo}$ * (L_t + D_t)
- $E_{arquivo}$: $N_{blocos\_arquivo}$ * (E_t + C_t)
- $R_{arquivo}$: T_A = E_t ou E_t + A_t

Ameaça:

Como é método envolve inicialmente somente a sobrescrita de uma unidade de apagamento, ele é extremamente rápido. Entretanto, os blocos do arquivo não são removidos, e, pelo menos em termos teóricos, podem vir a ser decifrados em um tempo futuro. Além disto, a operação de apagamento somente irá ser realizada quando todas as demais chaves da mesma unidade de apagamento forem apagadas, o que resulta em um tempo indeterminado para que isto ocorra.

5.4.8 Remoção pelo Método Proposto

O método híbrido de Sun (Seção 4.3 e análise na Seção 5.4.4) não considera que uma unidade de apagamento possa ter blocos livres, ou seja, blocos que ainda não foram utilizados em operações de escrita. Na análise que Sun realiza do seu método, ele somente considera blocos a serem removidos e blocos válidos. Isto causa problemas no cálculo do custo do apagamento ($Custo_A$) se não existem blocos válidos e a unidade de apagamento somente contiver blocos a serem removidos e blocos livres. Nesta situação, o método de Sun considera que o $Custo_A$ é zero. Para considerar que blocos livres serão apagados antes de serem escritos é um desperdício e acelera o desgaste da *flash*, propõe-se introduzir um fator de penalidade, proporcional ao número de blocos livres:

$$Penalidade_A(i) = N_{blocos\_livres} * E_t$$

Este fator indica que se perdeu a oportunidade de fazer N_{blocos_livres} escritas.

De forma semelhante, o método de Sun também tem problemas no cálculo do custo de sobrescrita ($Custo_{zo}$). Os blocos que são sobrescritos (com zeros) vão precisar passar por uma operação de apagamento no futuro, antes de poderem ser usados novamente. Assim, propõe-se introduzir um fator de penalidade proporcional ao número de blocos sobrescritos:

$$Penalidade_{zo}(i) = (N_{blocos\_a\_remover}\ (i) / N_{blocos\_unidade}\ ) * A_t$$

Com estas alterações, o método de Sun continua sendo realizado da mesma forma, mas o custo de cada operação é acrescido de um fator de penalidade:

Custo = Tempo – Benefício + Penalidade

As mesmas penalidades de apagamento e sobrescrita também poderiam ser aplicadas aos métodos de J. Lee (Seção 4.4) e Huang (Seção 4.5), pois ao realizarem operações de apagamento sobre unidades que contenham blocos livres estes métodos também aceleram o desgaste da mídia *flash*.

5.4.9 Conclusão da análise

Pelas características peculiares de uma memória *flash*, que necessita de operações especiais de apagamento, os métodos descritos nas seções anteriores levam em conta estas características quando realizam a remoção segura de um arquivo da memória *flash*. Embora todos sejam eficazes (exceto a remoção teórica por sobrescrita, incluída na análise somente como referencial), os métodos apresentam desempenho distinto e também afetam a durabilidade da memória *flash* de forma distinta (causando um desgaste prematuro do meio ao usarem operações de sobrescrita ou de apagamento de forma excessiva).

Remoção de arquivos por apagamento criptográfico é atrativa em termos de eficiência, mas como os blocos do arquivo permanecem na mídia até serem reciclados, sempre existe a ameaça de uma decifragem futura, seja, por recuperação da chave (se existirem cópias de reserva ou de custódia), seja por fraquezas do algoritmo ou da chave utilizada. Por outro lado, realizar sobrescrita e/ou apagamento em todos os blocos do arquivo causa um desgaste de todos estes blocos, além de deixar a operação de remoção de um arquivo bem mais demorada.

Ao serem analisados, identificou-se nos métodos pontos a serem melhorados, como o aprimoramento da função custo no método de Sun, e a consideração da existência de blocos livres (ver Seção 5.4.8)

6 SIMULADOR

Para realizar uma análise quantitativa e qualitativa dos métodos descritos no Capítulo 5, bem como uma análise dos aperfeiçoamentos propostos foi implementado um simulador na linguagem de programação C. Este simulador foi desenvolvido com as seguintes características:

- Simulação básica de um sistema de arquivos.
- Simulação das operações elementares de criação e remoção de arquivos.
- Sistemas de arquivos idealizado, independente de implementações práticas existentes.
- Simulação das operações da FTL : mapeamento de blocos virtuais para físicos, gerência do estado de blocos físicos, *garbage collector.*
- Simulação das operações sobre blocos físicos: leitura, escrita, sobrescrita e apagamento.
- Simulação dos métodos de remoção de arquivos do Capítulo 5. Observe-se que os métodos de apagamento criptográfico não foram implementados, pois apresentam custo e desempenhos constantes, independentes do tamanho ou da localização física do arquivo.

Essas características são descritas nas próximas seções. Uma visão geral do Simulador pode ser vista na Figura 13. O simulador recebe como entrada uma lista de comandos de criação e remoção de arquivos, e produz como resultado tanto uma imagem de memória com a situação após a execução destes comandos como uma série de relatórios sobre as operações de leitura, escrita e apagamento realizadas.

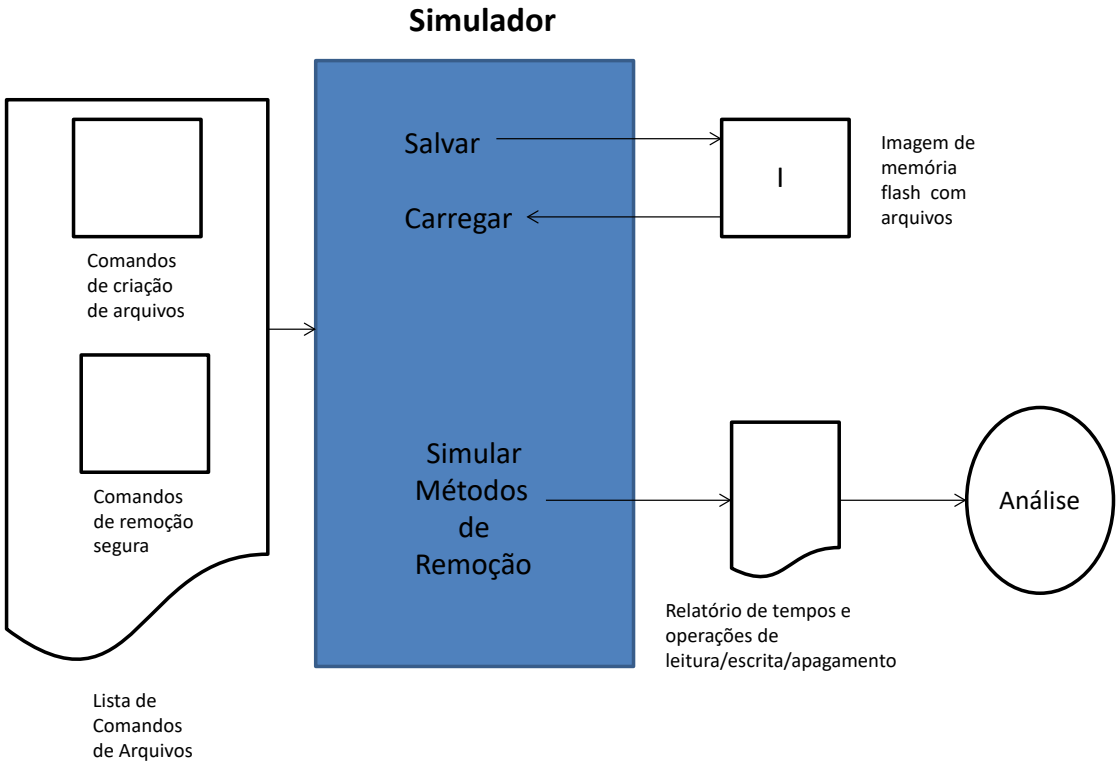


Figura 13 – Visão geral do simulador

6.1 **Sistema de arquivos**

O sistema de arquivos é constituído de duas partes distintas: um conjunto de arquivos e uma estrutura de diretório [48, Seção 11.1]. Como o simulador é desenvolvido para analisar as operações realizadas pela FTL, que somente opera sobre blocos virtuais e físicos, o sistema de arquivos simulado foi simplificado para somente modelar o conjunto de arquivos.

O sistema operacional abstrai as propriedades físicas de dispositivos de armazenamentos e define uma unidade lógica de armazenamento, o arquivo. Arquivos são mapeados, pelo sistema operacional, em dispositivos físicos que são normalmente não voláteis. Um arquivo recebe um nome para conveniência do usuário, mas, além disto, tem outros atributos, que variam de um sistema operacional para outro, mas normalmente consiste de [48, Seção 11.1.1]:

- Nome: nome simbólico em forma legível pelo usuário.
- Tipo: somente necessário em sistemas que suportam vários tipos.
- Localização: um ponteiro para um dispositivo e para a posição do arquivo neste dispositivo.
- Tamanho: o tamanho atual do arquivo, em bytes, palavras ou blocos.
- Proteção: informação de controle de acesso, sobre quem pode fazer qual operação sobre o arquivo.
- Data e Hora: Informação de tempo sobre a criação, última modificação e último uso do arquivo.

A informação sobre todos os arquivos é mantida em uma estrutura de diretório. Como a FTL somente opera sobre blocos e não lida diretamente com arquivos, para o desenvolvimento do simulador abstraiu-se a estrutura de diretórios e representou-se um arquivo somente por três atributos: Nome, tamanho em bytes e conjunto de blocos virtuais que formam o arquivo (detalhes na Seção 6.8).

6.2 **Operações sobre arquivos**

Um arquivo é um tipo abstrato de dados. Para definir adequadamente arquivos, também é necessário considerar as seis operações básicas que podem ser realizadas sobre eles [48, Seção 11.1.2].

- Criação de um arquivo: alocar espaço no dispositivo físico e criar uma entrada no diretório.
- Escrita em um arquivo: escrita de informação nos blocos que compõem o arquivo.
- Leitura de um arquivo: leitura da informação armazenada nos blocos que compõem o arquivo.
- Reposicionamento: determinação da posição do arquivo sobre a qual será realizada a próxima operação (*file seek*).
- Remoção de um arquivo: liberação da entrada do diretório e do espaço ocupado pelo arquivo no dispositivo físico.
- Trucagem: operação realizada quando se deseja manter os atributos do arquivo mas apagar o seu conteúdo. Evita remover um arquivo e recriá-lo.

Como o simulador é desenvolvido para analisar os diversos métodos de remoção segura, somente as operações de criação e de remoção de arquivo foram implementadas no simulador.

6.3 **Operações sobre blocos**

A FTL mapeia blocos virtuais para blocos físicos, e assim operações que o sistema operacional realiza sobre blocos virtuais são na realidade efetuadas sobre blocos físicos. O circuito que realiza estas operações é o MTD (Dispositivo de Tecnologia de Memória), como descrito na Seção 3.2. Este circuito é responsável pelas operações de leitura de blocos, escrita de blocos e apagamento de unidades.

No simulador, as operações de leitura, escrita e apagamento são modeladas pelos seus tempos de execução. O simulador contabiliza a quantidade de operações realizadas e o seus respectivos tempos, pois estes são os parâmetros necessários para a análise dos métodos de remoção. Não é necessário simular dados sendo realmente lidos, escritos ou apagados.

Observe-se que uma operação de sobrescrita, utilizada por alguns dos métodos de remoção, corresponde na realidade a uma operação de escrita, e é portanto modelada no simulador da mesma forma que uma escrita.

6.4 **Garbage Collector**

A FTL também é responsável por reciclar os blocos obsoletos, realizando uma operação de apagamento sobre toda a unidade. Este procedimento deve ser realizado sempre que esgotar a quantidade de blocos livres, disponíveis para escrita. Também pode ser ativado pela FTL quando não interferir com as operações de entrada e saída do sistema operacional, como o comando TRIM da Intel [20].

No simulador foram implementadas quatro estratégias de reciclagem:

- TRIM: recicla uma unidade de apagamento se todos os seus blocos estiverem marcados como obsoletos.
- Leve: recicla uma unidade de apagamento se todos os seus blocos estiverem marcados como obsoletos ou válidos (não existem blocos livres), e a quantidade de blocos obsoletos é maior que a quantidade de blocos válidos. Os blocos válidos são copiados para outra unidade antes de realizar o apagamento.
- Média: recicla uma unidade de apagamento se todos os seus blocos estiverem marcados como obsoletos ou válidos (não existem blocos livres), e existe no mínimo um bloco obsoleto. Os blocos válidos são copiados para outra unidade antes de realizar o apagamento.
- Agressiva: recicla uma unidade de apagamento se no mínimo um de seus blocos estiver marcado como obsoleto, independente da quantidade de blocos válidos ou livres. Os blocos válidos são copiados para outra unidade antes de realizar o apagamento. Como os blocos livres são apagados antes de poderem ser usados, esta estratégia desperdiça estes blocos, e contribui para o maior desgaste da mídia. Somente deve ser usada se não houver nenhuma outra possibilidade de reciclar blocos.

6.5 **Métodos de remoção**

Foram implementados no simulador os métodos descritos no Capítulo 5 e algumas variações decorrentes da análise realizada na Seção 5.4.8. Resumidamente, estes métodos são:

- Remoção normal: somente marcação do bloco como obsoleto
- Remoção exclusivamente por sobrescrita com zeros (ZO, ou *zero-overwrite*): fornece o limite inferior do número de operações necessárias
- Remoção exclusivamente por apagamento (A) de unidades de apagamento: fornece o limite superior da quantidade de operações de apagamento necessárias
- Método de Sun, que compara custos de sobrescrita com zeros (ZO) e de apagamento (A), mas considera um Benefício (B) no custo (Seção 5.4.4) ao realizar apagamentos (no simulador, identificado pela sigla ZO-AB)
- Método de Sun modificado, sem considerar Benefício no apagamento (no simulador, identificado pela sigla ZO-A)
- Método proposto: método de Sun considerando um custo de Penalidade (Seção 5.4.8) tanto para sobrescrita com zeros como para apagamento (identificado pela sigla ZOP-ABP)
- Método proposto: método de Sun com Penalidade somente para apagamento (sigla ZO-ABP)
- Método proposto: método de Sun com Penalidade para sobrescrita e apagamento, mas sem considerar o Benefício no apagamento (sigla ZOP-AP)
- Método proposto: método de Sun com Penalidade somente para apagamento, e sem considerar Benefício para este apagamento (sigla ZO-AP)
- Método proposto: método de Sun com Benefício para apagamento e Penalidade para sobrescrita com zeros (sigla ZOP-AB)
- Método proposto: método de Sun sem Benefício para Apagamento e com Penalidade para sobrescrita com zeros (sigla ZOP-A)
- Método de Huang, que realiza tanto sobrescrita com zeros como apagamento, e portanto não necessita de funções de custo para decisão entre estas operações

A Tabela 7 apresenta um quadro com os métodos propostos em função da aplicação ou não da função benefício (B) no custo de um apagamento e das funções de Penalidade (P) nos custos de apagamento e sobrescrita com zeros. Os números em parênteses indicam o índice do método proposto nas funções internas do simulador.

Tabela 7 – Métodos de cálculo de custo de apagamento e sobrescrita

	Apagamento	Apagamento com Benefício	Apagamento com Penalidade	Apagamento com Benefício e Penalidade
Sobrescrita (*Zero-Overwrite*)	ZO-A (0)	ZO-AB (Sun)	ZO-AP (4)	ZO-ABP (2)
Sobrescrita com Penalidade	ZOP-A (6)	ZOP-AB (5)	ZOP-AP (3)	ZOP-ABP (1)

Observe-se que não é definida nenhuma função de benefício no custo de uma operação de sobrescrita com zeros. Assim, a operação de sobrescrita pode ser calculada de forma simples (ZO) ou com penalidade (ZOP), enquanto que a função de apagamento pode ser aplicada de forma simples (A), com a função de benefício (AB), com a função de penalidade (AP) ou com ambas (ABP)

6.6 **Modelagem de uma Memória *Flash***

Para o simulador, uma memória *flash* é caracterizada pelo tamanho em bytes dos seus elementos internos (blocos e unidades de apagamento) e pelos tempos de suas operações (leitura, escrita e apagamento). Assim, para modelar uma memória *flash* no simulador foram utilizados os seguintes parâmetros:

- BlockSize: tamanho, em Bytes, de um bloco físico. Tipicamente varia entre 2048 e 16384 Bytes.
- EUSize: tamanho, em Bytes, de uma unidade de apagamento (*erase unit*). É um múltiplo do tamanho de um bloco físico, e tipicamente contém de 32 a 256 blocos.
- Blocks_per_EU: número de blocos físicos em uma unidade de apagamento. Este valor é calculado automaticamente pelo simulador, em função de BlockSize e EUSize.
- VirtualBlockSize: tamanho, em Bytes de um bloco virtual. Sempre tem o mesmo tamanho que um bloco físico. O acesso do sistema operacional à memória é realizado através de blocos virtuais, que a FTL mapeia para blocos físicos.
- FlashSize: tamanho total da *flash*, expresso em quantidade de blocos físicos. Como o simulador foi desenvolvido para testar e analisar métodos de remoção, esta tamanho não é um parâmetro crítico para o simulador, mas deve ser grande o suficiente para conter uma quantidade significativa de arquivos.
- ClusterSize: unidade básica de alocação utilizada pelo sistema operacional quando necessita de espaço para um arquivo [48, Seção 11.1.5]. Esta unidade de alocação é caracterizada pelo seu tamanho em Bytes, e é constituída de um número inteiro de blocos.
- Blocks_per_Cluster: número de blocos em uma unidade de alocação (*cluster*). Este valor é calculado automaticamente pelo simulador, em função de ClusterSize e BlockSize.
- DirectorySize: o simulador foi modelado para ser independente de um sistema de arquivos, mas é necessária uma estrutura de dados que descreva os atributos dos arquivos sendo criados e removidos (ver Seção 6.8). Este parâmetro determina o tamanho desta estrutura, e foi dimensionado para ser grande o suficiente para conter todos os arquivos sendo simulados.
- ReadTime: tempo, em microssegundos, de leitura de um bloco físico da *flash*. Varia tipicamente de 10 a 250 microssegundos.
- WriteTime: tempo, em microssegundos, de escrita de um bloco físico na *flash*. Varia tipicamente de 50 a 2700 microssegundos.
- EraseTime: tempo, em microssegundos, para realizar o apagamento de uma unidade da *flash*. Varia tipicamente de 2000 a 6000 microssegundos.

A Tabela 8, retirada de [54] e de [27] ilustra tamanhos típicos de blocos e de unidades de apagamento.

Tabela 8 – Tamanhos típicos de blocos e unidades de apagamento

Tamanho do Bloco	Tamanho da Unidade de Apagamento	Blocos por Unidade	Capacidade da Memória *Flash*
512 Bytes	16 KBytes	32	64 MBytes
2048 Bytes	128 Kbytes	64	128 a 256 Mbytes
4096 Bytes	256 KBytes	64	512 Mbytes a 4 Gbytes
4096 Bytes	512 Kbytes	128	2 a 8 Gbytes
8192 Bytes	1 Mbytes	128	4 a 64 GBytes

A Tabela 9, retirada de [27], [54] e [35] ilustra tempos típicos das operações de leitura, escrita e apagamento.

Tabela 9 – Tempos típicos de leitura, escrita e apagamento

	Leitura	Escrita	Apagamento
Kingston UHS	10 μs	50 μs	3 ms
Toshiba SLC	30 μs	250 μs	3 ms
Toshiba MLC	250 μs	2700 μs	4 ms
Micron MT29F2A	25 μs	220 μs	500 μs
Micron MT29F2WP	25 μs	300 μs	2 ms
Micron MT29F8A	25 μs	220 μs	1,5 ms

Os parâmetros descritos acima podem variar, para permitir analisar o desempenho dos diversos métodos de remoção em função destes parâmetros. Os parâmetros são lidos de um arquivo de configuração (simulator.ini), quando o simulador é inicializado. A Figura 14 mostra um exemplo deste arquivo de configuração. Os blocos são medidos em Bytes, enquanto os tempos são medidos em microssegundos. Se o arquivo de inicialização não é encontrado, o simulador assume valores padrão para estes parâmetros.

```
4096      ; BlockSize
262144    ; EUSize
4096      ; VirtualBlockSize
4096      ; ClusterSize
10        ; ReadTime
50        ; WriteTime
3000      ; EraseTime
```

Figura 14 – Exemplo Arquivo de configuração do simulador

6.7 **Comandos do Simulador**

Para permitir simular e analisar os métodos de remoção segura, foram implementados no simulador os seguintes comandos:

- t: TRIM – Realiza uma operação manual de TRIM para reciclar unidades de apagamento.

- g: Garbage Collection(método) – realiza uma reciclagem manual, baseado no método especificado (leve, média, agressiva). Como método também pode ser especificado “info”, onde a reciclagem não é efetuada, e somente retorna informação de quantas unidades de apagamento seriam recicladas.
- n: Novo arquivo(nome, tamanho) – cria um novo arquivo utilizando o nome e o tamanho em Bytes fornecidos. Ocupa blocos virtuais e físicos em caso de sucesso, e retorna erro caso já exista um arquivo com o nome especificado ou não exista espaço suficiente na mídia para criar o arquivo.
- m: Método padrão de remoção(método) – determina qual o método de remoção a ser utilizado em futuras remoções. Os métodos suportados são os listados na Seção 6.5.
- d: Deleta arquivo(nome) – remove um arquivo, utilizando o método de remoção padrão. Retorna erro não exista arquivo com o nome especificado.
- s: Salva imagem(arquivo) – salva uma imagem do estado atual da mídia no arquivo especificado (ver Seção 6.7 para informação sobre a imagem).
- l (load): Carrega imagem(arquivo) – carrega uma imagem previamente salva no arquivo especificado.
- p (print): Imprime relatório(arquivo) – salva informação sobre o estado atual da mídia no arquivo texto especificado.
- r (record): registro(arquivo) – salva no arquivo texto especificado um relatório de todas as operações realizadas sobre a mídia, inclusive com detalhes sobre os métodos de remoção.
- x (xls): contabilizar(arquivo) – salva no arquivo especificado informações sobre as operações realizadas em formato CVS, para ser posteriormente processado por planilha eletrônica. São contabilizadas as operações realizadas sobre blocos quando da remoção de um arquivo, envolvendo leitura (read), escrita (write), sobrescrita com zero (zero-overwrite), apagamento (erase) e marcação como inválido (obsolete). Para fins de análise, também são contabilizados a quantidade de blocos livres que foram prematuramente apagados, assim como a quantidade total de blocos envolvidos na remoção de arquivos.
- e (end): encerra processamento.

6.8 **Imagem de memória**

Um arquivo de imagem de memória contém informação sobre o estado atual da mídia sendo simulada. Este arquivo é dividido em cinco seções, conforme ilustrado na Figura 15.

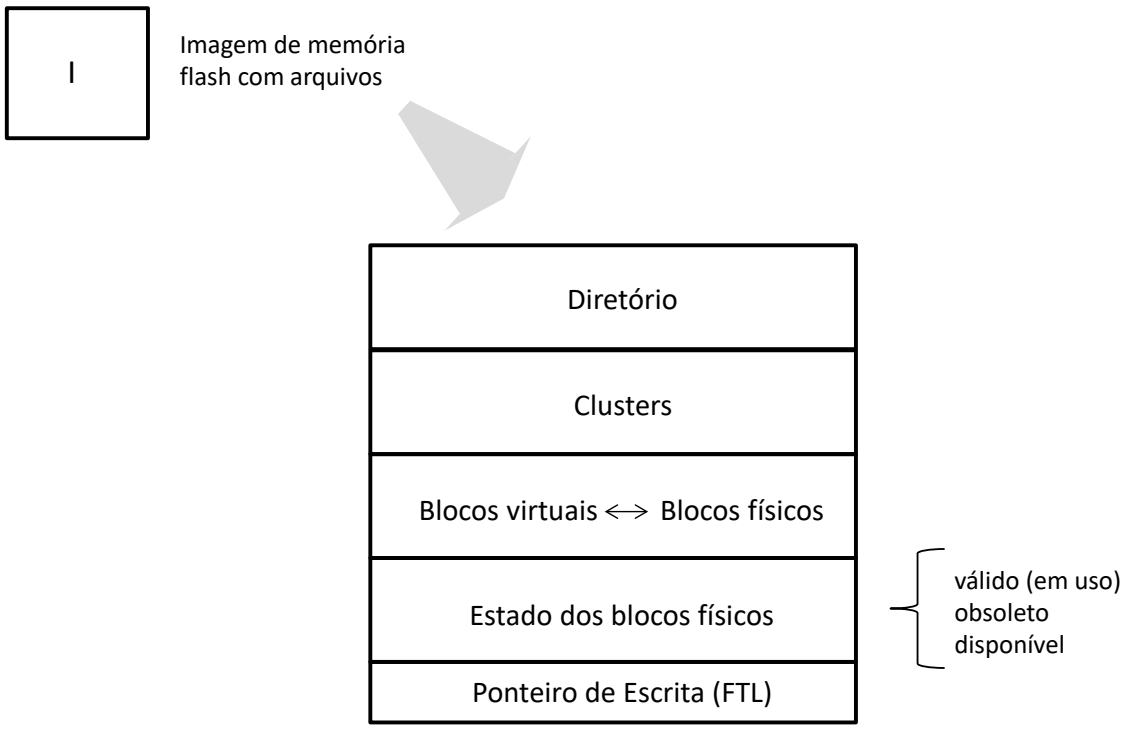


Figura 15 – Seções de uma imagem de memória *flash*

As seções de Diretório e Clusters simulam de forma simplificada as informações básicas de um sistema de arquivos, como pode ser visto na Figura 16.

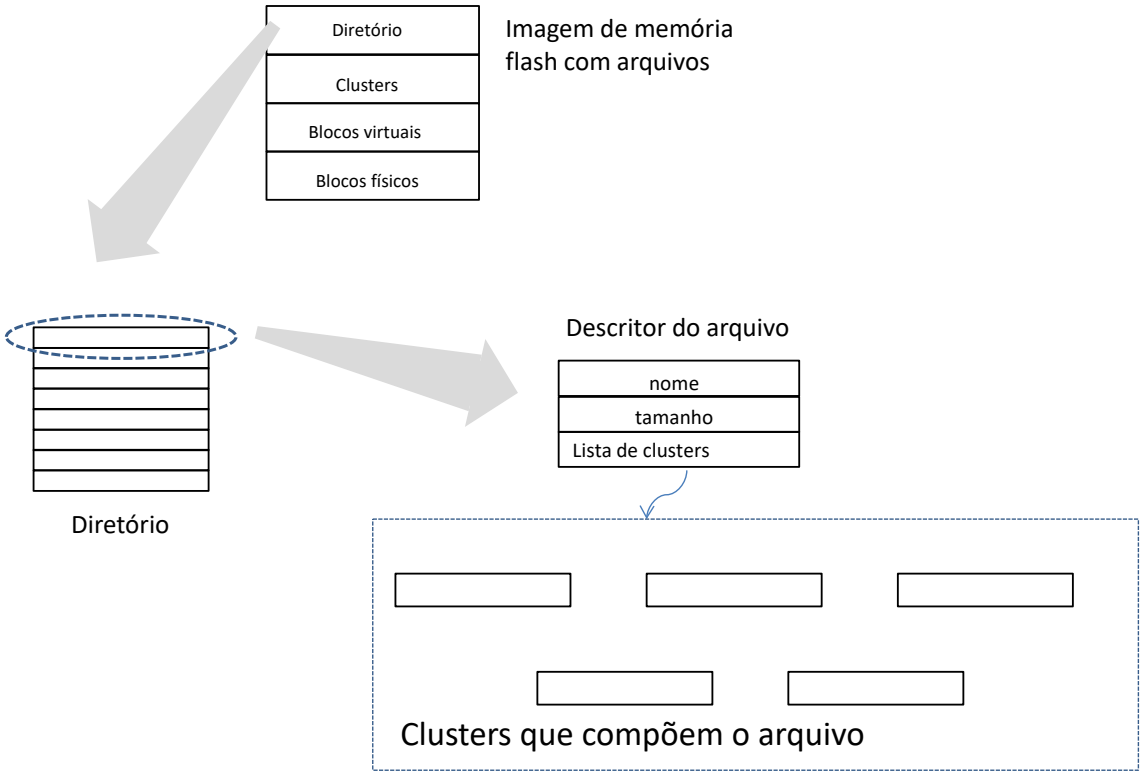


Figura 16 – Informações de arquivo na imagem de memória *flash*

Um arquivo é somente modelado pelo seu nome, seu tamanho (em Bytes) e pelo conjunto de clusters que ocupa na mídia. E o Diretório simulado é simplesmente uma lista de arquivos. Esta modelagem simples permite que o simulador opere independente de um sistema de arquivos,

uma vez que os métodos de remoção são analisados no nível da FTL, que somente opera com blocos virtuais e físicos e desconhece a noção de um sistema de arquivos (Seção 3.1).

A FTL e os blocos físicos da memória *flash* são descritos por três seções, que contém informação sobre o mapeamento, o estado dos blocos e o valor do ponteiro de escrita. A seção de mapeamento de blocos virtuais para físicos simula o mapeamento realizado pela FTL, como ilustrado na Figura 17.

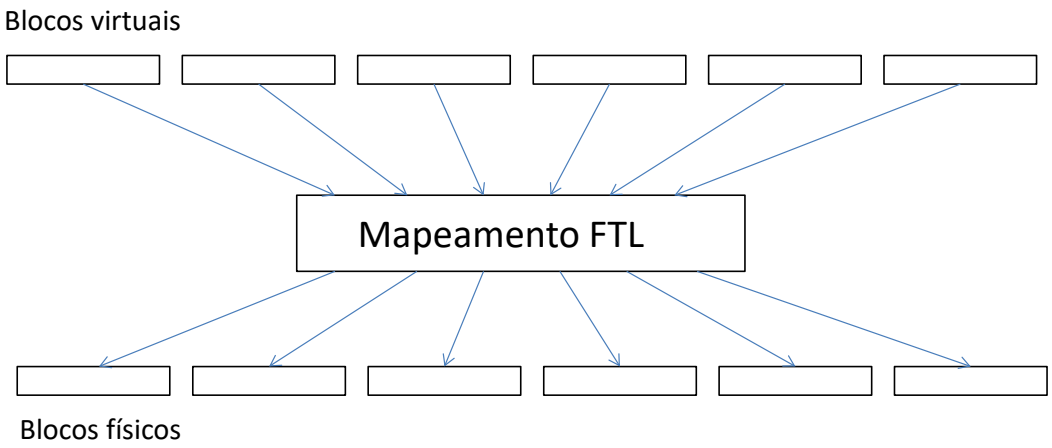


Figura 17 – Mapeamento de blocos virtuais para físicos na imagem de memória *flash*

A seção de estado dos blocos físicos indica, para cada bloco físico da mídia, qual o seu estado atual: disponível (livre), válido (em uso) ou obsoleto (marcado para reciclagem).

A seção do Ponteiro de Escrita indica qual o próximo bloco físico a ser utilizado em uma operação de escrita. Este ponteiro é empregado pela FTL para garantir um desgaste uniforma da mídia.

6.9 **Operação do Simulador**

O simulador recebe os comandos a partir de um arquivo texto, e fornece seus resultados em uma série de arquivos de saída, como ilustrado na Figura 13.

Um arquivo de entrada do simulador contém uma série de comandos para criar e remover arquivos, além de salvar e carregar arquivos de imagem de memória e gerar relatórios. A Figura 18 ilustra uma série de comandos. A Seção 6.7 descreve os detalhes de cada comando.

As duas primeiras linhas indicam que se deseja registrar um relatório em texto no arquivo Commands1a-log.txt e um relatório em planilha no arquivo Commands1a.xls. A seguir são criados alguns arquivos (comando n) e removidos dois destes n (comando d). Como o método de remoção não é especificado utiliza-se a remoção padrão (remoção normal). Após, cria-se uma imagem de disco (comando s disk1.txt) e um arquivo texto com a descrição legível desta imagem (comando p disk-info.txt). A seguir realiza-se uma operação de TRIM (comando t) e uma reciclagem de forma agressiva (comando g aggressive). Logo após criam-se uma nova imagem de disco (comando s disk1-a.txt) e um relatório descritivo desta imagem (comando p disk-a-info.txt). As três últimas linhas encerram o relatório em texto iniciado na linha 1 (comando r), o relatório em planilha iniciado na linha 2 (comando x) e a operação do simulador (comando e).

```
r Commands1a-log.txt
x Commands1a.xls
n Arquivo1.txt 12564
n Arquivo2.txt 78217
n Arquivo3.txt 225280
d Arquivo2.txt
n Arquivo4.txt 95783
n Arquivo5.txt 1024
n Arquivo6.txt 65000
n Arquivo7.txt 65537
d Arquivo5.txt
n Arquivo8.txt 48000
s disk1.txt
p disk1-info.txt
t
g aggressive
s disk1-a.txt
p disk1-a-info.txt
r
x
e
```

Figura 18 – Exemplo simples de comandos para o simulador

A Figura 19 mostra o conteúdo do arquivo de relatório Commands1a-log.txt e a Tabela 10 mostra a planilha com a quantidade de operações realizadas nas remoções dos dois arquivos.

```
File Arquivo1.txt created successfully
File Arquivo2.txt created successfully
File Arquivo3.txt created successfully
Deleting (unlinking) file Arquivo2.txt in File System
Normal Delete: 20 blocks marked as obsolete
File Arquivo2.txt deleted successfully
File Arquivo4.txt created successfully
File Arquivo5.txt created successfully
File Arquivo6.txt created successfully
File Arquivo7.txt created successfully
Deleting (unlinking) file Arquivo5.txt in File System
Normal Delete: 1 blocks marked as obsolete
File Arquivo5.txt deleted successfully
File Arquivo8.txt created successfully
Image file disk1.txt saved successfully
Report file disk1-info.txt saved successfully
Erase Units recycled by TRIM: 0 (0 blocks)
Garbage Collection status: 0 trim, 0 light, 2 mild, 2 aggressive
Performed aggressive garbage collection. Recovered 2 EU (128 blocks)
Image file disk1-a.txt saved successfully
Report file disk1-a-info.txt saved successfully
```

Figura 19 – Exemplo de relatório de uma simulação

A Tabela 10 mostra a quantidade de operações realizadas sobre os blocos da memória *flash* durante a remoção dos dois arquivos. Como foi utilizado o método de remoção normal, simplesmente os blocos ocupados pelos arquivos foram marcados como obsoletos, para serem futuramente reciclados. Não houve a necessidade de sobrescrever ou apagar nenhum bloco.

Tabela 10 – Exemplo de contabilização das operações realizadas em uma simulação

Arquivo	Método	Leituras	Escritas	Apagamentos	Sobrescritas com zeros	Marcados obsoletos	Livres apagados	Blocos operados
Arquivo2.txt	Normal	0	0	0	0	20	0	0
Arquivo5.txt	Normal	0	0	0	0	1	0	0
	total	0	0	0	0	21	0	0

A Figura 20 ilustra um exemplo mais complexo, onde três arquivos são criados, uma imagem de disco é salva e após seis métodos de remoção utilizados (remoção normal, por sobrescrita ou *zero-overwrite*, por apagamento, pelo método de Sun, pelo método proposto na Seção 5.4.8 e pelo método de Huang). Antes de remover por cada um dos métodos, a imagem de disco gerada é recarregada, para que todos os métodos operem sobre a mesma imagem de disco. Após cada remoção o estado do disco é impresso para análise posterior.

```
r disk-test1-log.txt
n Arquivo1.txt 12564
n Arquivo2.txt 78217
n Arquivo1.txt 225280
s disk-test1.txt
x disk-test1.xls
l disk-test1.txt
d Arquivo1.txt n
p disk-test1-normal.txt
l disk-test1.txt
d Arquivo1.txt z
p disk-test1-zero-over.txt
l disk-test1.txt
d Arquivo1.txt e
p disk-test1-erase.txt
l disk-test1.txt
d Arquivo1.txt s
p disk-test1-sun.txt
l disk-test1.txt
d Arquivo1.txt w
p disk-test1-jw.txt
l disk-test1.txt
d Arquivo1.txt h
p disk-test1-huang.txt
r
x
e
```

Figura 20 – Exemplo de simulação comparativa entre os métodos de remoção

A Tabela 11 mostra uma análise comparativa entre os seis métodos de remoção simulados pelos comandos da Figura 20, extraído do relatório gerado pelo simulador no arquivo disk-test1.xls (planilha eletrônica em formato cvs).

Tabela 11 – Análise comparativa das operações realizadas por diversos métodos de remoção

Arquivo	Método	Leituras	Escritas	Apagamentos	Sobrescritas com zeros	Marcados obsoletos	Livres apagados	Blocos operados
Arquivo1.txt	Normal	0	0	0	0	4	0	0
Arquivo1.txt	Zero-over	0	0	0	4	4	0	4
Arquivo1.txt	Erase	20	20	1	0	0	40	104
Arquivo1.txt	Sun	0	0	0	4	4	0	4
Arquivo1.txt	Proposto	0	0	0	4	4	0	4
Arquivo1.txt	Huang	20	20	1	4	0	40	108
	total	40	40	2	16	4	80	224

O método normal somente marcou os 4 blocos do arquivo como inválidos (obsoletos) para serem reciclados futuramente. O método de remoção por sobrescrita realizou uma sobrescrita com zeros nestes mesmos 4 blocos. Já o método de apagamento realizou 20 operações de leitura e escrita, para copiar os blocos válidos (de outros arquivos) para outra unidade antes de apagar a unidade onde estava o arquivo. Tanto o método de Sun como o método proposto realizam 4 sobrescritas, pois o custo associado é menor que o custo de um apagamento. Finalmente, o método de Huang realiza tanto 4 sobrescritas como as 20 leituras e cópias necessárias para o apagamento da unidade onde se encontra o arquivo.

Como um dos objetivos do Simulador é realizar uma análise comparativa entre os diversos métodos de remoção, ele foi programado para poder receber um arquivo e executar seus comandos (modo *single*) e para poder receber um arquivo de comandos e executar estes comandos sobre todos os métodos implementados (modo *multiple*).

7 EXPERIMENTOS

Com o Simulador especificado no Capítulo 6 foram realizados diversos experimentos, para avaliar os métodos descritos na literatura e analisados no Capítulo 5 e também para análise de aprimoramento do método escolhido. Com o uso do Simulador foi possível validar estes métodos em diferentes cenários de uso e com arquivos de diversos tamanhos. Para todas as simulações realizadas foram utilizados doze cenários distintos, que efetuavam a remoção de um total de 2.437 arquivos. Detalhes sobre estes cenários e os resultados das simulações podem ser encontrados no Apêndice A.

7.1 **Análise Inicial dos Métodos**

Na primeira versão do simulador foram implementados 6 métodos de remoção: normal, por sobrescrita de zeros, por apagamento, pelo método de Sun (Seção 5.4.4), pelo método proposto (Seção 5.4.8) e pelo método de Huang (Seção 5.4.6).

A remoção normal, executada pelo sistema operacional, somente libera os blocos utilizados pelo arquivo. Esta liberação dos blocos virtuais significa que a FTL irá marcar os blocos físicos equivalentes como obsoletos (inválidos), ou seja, podem ser apagados na próxima reciclagem. Observe-se que os blocos físicos continuam com o seu conteúdo original, e podem ser recuperados por métodos de forense computacional.

A remoção por sobrescrita de zeros também marca os blocos físicos do arquivo como obsoletos, mas adicionalmente substitui os seus conteúdos com zeros, impedindo assim sua recuperação posterior. Desta maneira, o custo deste método de remoção corresponde a tantas operações de escrita quantos forem os blocos do arquivo. Assim, o custo da remoção por sobrescrita pode ser representado por:

$$Custo_{sobrescrita} = T_{sobrescrita} = N_{blocos\_arquivo} * E_t$$

A remoção por apagamento realiza uma operação de apagamento (*erase blocks*) na unidade de apagamento (*erase unit*) onde os blocos físicos estiverem localizados. Como esta unidade de apagamento também pode conter blocos de outros arquivos, estes blocos válidos devem ser previamente copiados para outra unidade de apagamento, antes da unidade atual poder ser apagada. Isto implica em uma série de operações de leitura e escrita (tantas quantas forem os blocos válidos a serem movidos) e posteriormente em uma operação de apagamento. Comparada com a remoção por sobrescrita de zeros, esta remoção por apagamento é potencialmente de maior custo, mas apresenta a grande vantagem da unidade de apagamento ficar pronta para uso imediato e não depender de uma reciclagem futura. Por outro lado, a realização de um apagamento de forma imediata acelera o desgaste do meio físico. O custo da remoção por apagamento implica em apagar todas as unidades de apagamento que contêm blocos do arquivo:

$$Custo_{Apagamento} = T_A total = \Sigma\, T_A(i),$$

onde para cada unidade de apagamento $T_A(i) = N_{blocos\_válidos}(i) * (L_t + E_t) + A_t$

Comparando-se a remoção por sobrescrita de zeros com a remoção por apagamento, e utilizando-se como fator custo unicamente o tempo de execução das operações necessárias, obtém-se o gráfico da Figura 21. Para este gráfico, considerou-se um tempo de leitura de bloco como 10 µs, um tempo de escrita de 50 µs e um tempo de apagamento de 3 ms (3000 µs), além de uma unidade de apagamento com 64 blocos físicos.

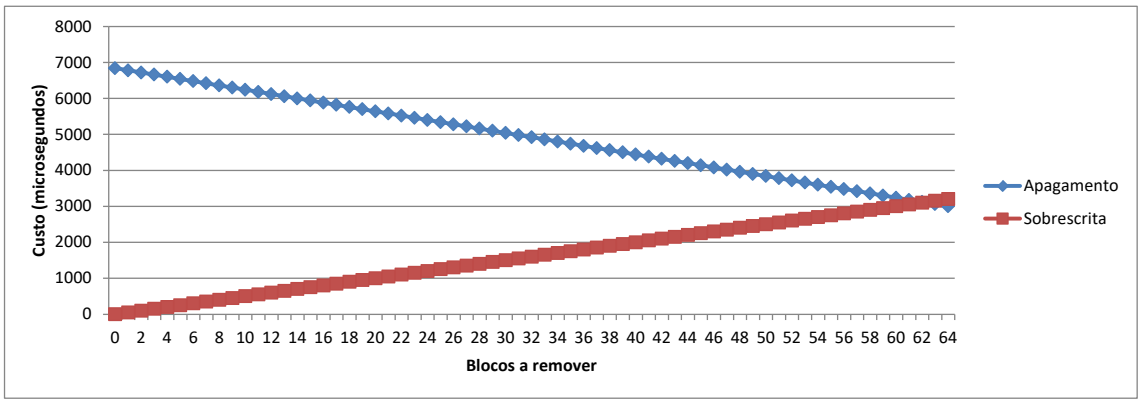


Figura 21 – Comparação de tempos de sobrescrita e apagamento

Observe-se na Figura 21 que um apagamento somente apresenta vantagem quando o número de blocos a remover se aproxima da quantidade de blocos em uma unidade de apagamento. Como um apagamento tem a vantagem de dispensar uma futura reciclagem dos blocos, Sun [51] propôs incluir esta característica na forma de um benefício (Seção 5.4.4):

$$Benefício_A(i) = (N_{blocos\_ganhos}(i) / N_{blocos\_unidade}) * A_t$$

onde $N_{blocos\_ganhos}(i) = N_{blocos\_unidade} - N_{blocos\_válidos}(i)$

ou $Benefício_A = (1 - (N_{blocos\_válidos}(i) / N_{blocos\_unidade})) * A_t$

O efeito deste benefício pode ser visto na Figura 22. Conforme o número de blocos a remover se aproxima da quantidade de blocos da unidade, menor o custo de um apagamento. Observe-se que, pela subtração do Benefício no Custo do Apagamento, este Custo é reduzido a zero quando devem ser removidos todos os blocos de uma unidade (o que seria feito por uma operação de reciclagem de qualquer maneira).

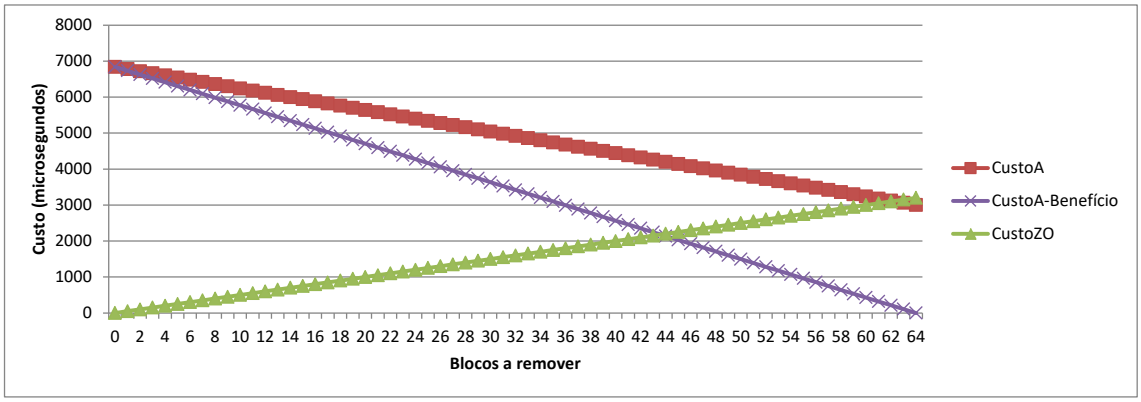


Figura 22 – Benefício definido por Sun para apagamento

Desta forma, para cada unidade de apagamento o método de Sun decide entre sobrescrita e apagamento através de uma função de custo, escolhendo aquela operação de menor custo. Como estes custos são função do número de blocos a remover e do número de blocos válidos, uma análise é feita para cada unidade de apagamento, escolhendo-se sempre a operação de menor custo para aquela unidade:

$$Custo_{zo}(i) = N_{blocos\_a\_remover}(i) * E_t$$

$$Custo_{A}(i) = N_{blocos\_válidos}(i) * ((L_t + E_t) + A_t / N_{blocos\_unidade})$$

$$Custo_{Sun} = \Sigma \min (Custo_{zo}(i), Custo_{A}(i))$$

A Figura 23 ilustra o impacto do método de Sun sobre os métodos de sobrescrita e apagamento. Como pode ser visto, se o número de blocos a remover for menor que 44, realiza-se uma sobrescrita, senão realiza-se um apagamento com o benefício de realizar simultaneamente a reciclagem da unidade de apagamento.

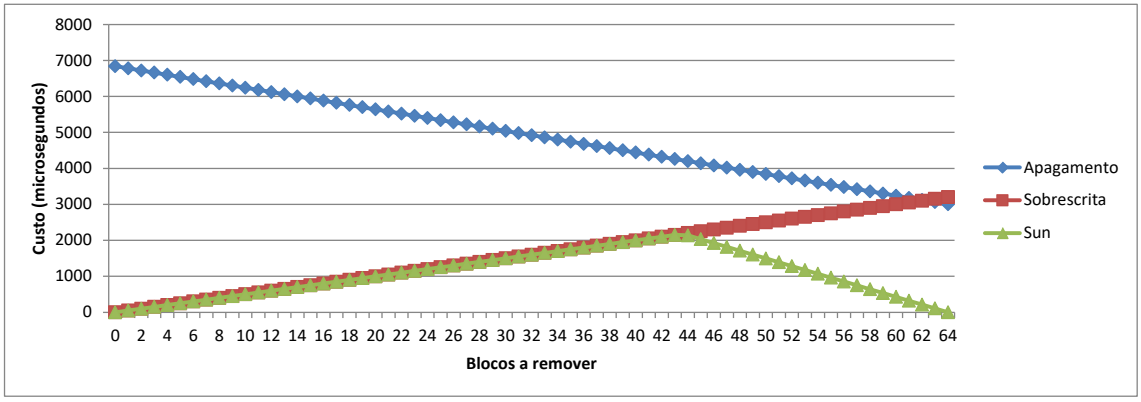


Figura 23 – Comparativo do método de Sun com sobrescrita e apagamento

Como os custos do método de Sun dependem dos tempos das operações de leitura, escrita e apagamento, o ponto de decisão entre sobrescrita e apagamento varia de uma memória *flash* para outra, mas para cada tipo de memória *flash* este ponto é constante para todas as unidades de apagamento, podendo ser claramente determinado em função do número de blocos a remover, como ilustrado na Figura 24. Nesta figura, os tempos são representados na forma leitura-escrita-apagamento, em microssegundos.

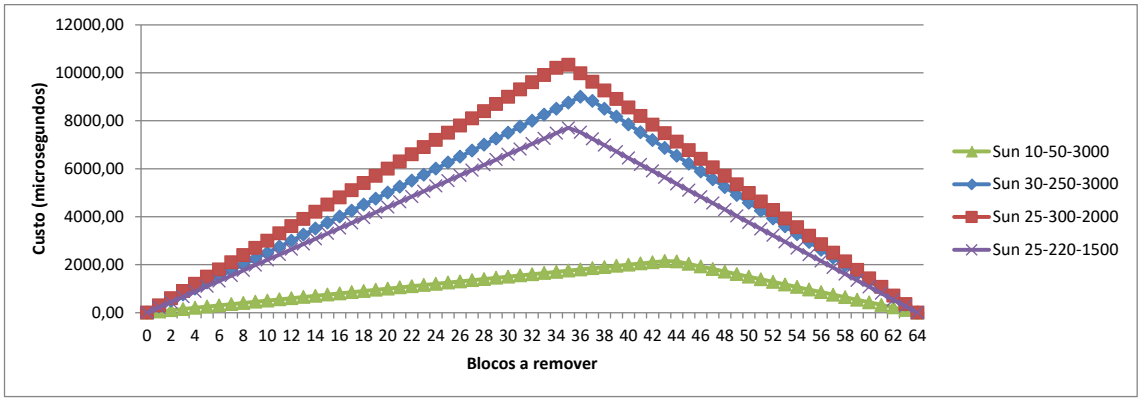


Figura 24 – Curvas de decisão do método de Sun

Entretanto, como analisado na Seção 5.4.8, o método de Sun calcula os custos somente em função dos blocos a remover e dos blocos válidos. Os custos não consideram a existência de blocos livres (*available*), ou seja, blocos de uma unidade de apagamento que ainda não foram utilizados em nenhuma escrita. Como estes blocos livres vão ser apagados antes de serem utilizados, ao custo de uma operação de apagamento deve ser acrescentado de um fator de penalidade:

$$Penalidade_A(i) = N_{blocos\_livres} * E_t$$

Este fator indica que se perdeu a oportunidade de fazer N_{blocos_livres} escritas, pois a unidade vai ser apagada antes disto.

O método de Sun também pode ser modificado no cálculo do custo de sobrescrita ($Custo_{zo}$). Os blocos que são sobrescritos (com zeros) vão precisar passar por uma operação de apagamento no futuro, antes de poderem ser usados de novo. Para refletir isto, propõe-se introduzir um fator de penalidade proporcional ao número de blocos sobrescritos:

$$Penalidade_{zo}(i) = (N_{blocos\_a\_remover}(i) / N_{blocos\_unidade}) * A_t$$

Com estas alterações, a modificação proposta no método de Sun adiciona ao custo de cada operação um fator de penalidade:

$$Custo = Tempo - Benefício + Penalidade$$

$$ou\ Custo_{zo}(i) = N_{blocos\_a\_remover}(i) * E_t + (N_{blocos\_a\_remover}(i) / N_{blocos\_unidade}) * A_t$$

$$Custo_A(i) = N_{blocos\_válidos}(i) * ((L_t + E_t) + A_t / N_{blocos\_unidade}) + N_{blocos\_livres} * E_t$$

A consequência da adição destas penalidades pode ser vista na Figura 25. Observe-se que na figura considera-se a existência de 32 blocos livres na unidade de apagamento, que é constituída de 64 blocos no total. O custo calculado por Sun realiza sobrescrita até 22 blocos a remover, e passa a realizar apagamento a partir de 23 blocos a remover. Com a adição das penalidades, a intersecção do custo de sobrescrita (CustoZO-32) com o custo de apagamento (CustoA-32) passa para 25 blocos a remover, ou seja, devem existir agora mais blocos a remover para que a operação de apagamento seja a escolhida. Desta forma procura-se retardar o apagamento devido à existência dos blocos livres.

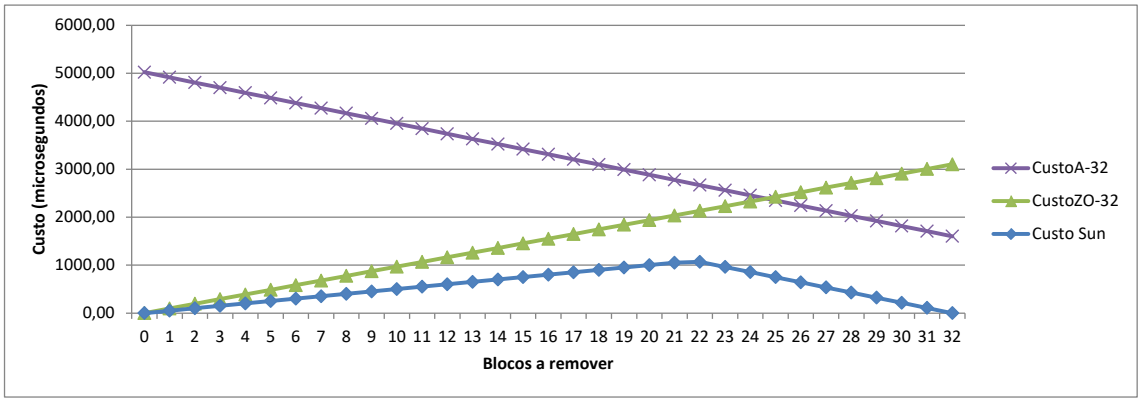


Figura 25 – Influência da adição de penalidades quando existem 32 blocos livres

Quanto maior for a quantidade de blocos livres, maior será a influência dos fatores de penalidade. Por exemplo, a Figura 26 ilustra o caso de 48 blocos livres em uma unidade de apagamento de 64 blocos, ou seja, existem no máximo 16 blocos a remover. Enquanto pelo método de Sun o ponto de escolha entre sobrescrita e apagamento está localizado em 11 blocos a remover, a adição de penalidades faz com que sempre seja realizada uma sobrescrita, independente da quantidade de blocos a remover. Isto faz sentido, pois um apagamento, mesmo no caso de 16 blocos a remover, iria também apagar 48 blocos livres que não foram utilizados, o que contribuiria para o aumento do desgate da memória.

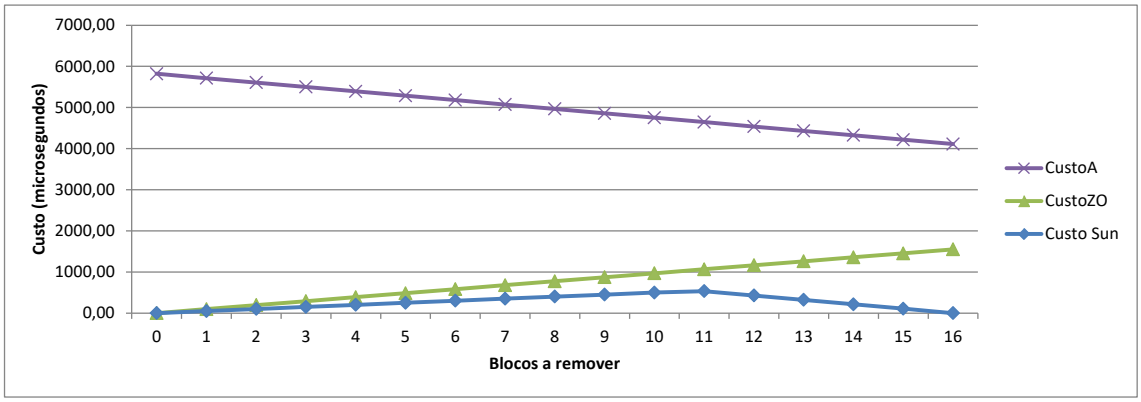


Figura 26 – Influência da adição de penalidades quando existem 48 blocos livres

Um método composto que utiliza tanto sobrescrita com zeros como apagamento foi proposto por Huang [17], para atender a norma de apagamento segura no NIST [40]. Assim, a remoção de um arquivo inicialmente sobrescreve os blocos destes arquivos e posteriormente apaga todas as unidade de apagamento que contêm blocos deste arquivo (com a cópia de blocos válidos para outra unidades):

$$Custo_{Huang} = \Sigma\, T_{zo}(i) + \Sigma\, T_A(i)$$

onde, para cada unidade de apagamento,

$$T_{zo}(i) = N_{blocos\_a\_remover}(i) * E_t$$

e $$T_A(i) = N_{blocos\_válidos}(i) * (L_t + E_t) + A_t$$

Os métodos descritos acima foram implementados no simulador e foi executada uma série de experimentos, que realizaram a remoção total de 2437 arquivos. Os resultados consolidados destes experimentos podem ser visto na Tabela 12. Para os experimentos realizados, foram empregados os mesmo tempos já discutidos nesta seção, ou seja, leitura de um bloco em 10 µs, escrita de um bloco em 50 µs e apagamento em 3000 µs, assim como 64 blocos por unidade de apagamento. Para cada método foram contabilizadas a quantidade de blocos copiados entre unidade de apagamento (leituras e escritas), a quantidade de unidades apagadas (apagamantos), a quantidade blocos sobrescritos (sobrescritas com zeros), a quantidade de blocos marcados com inválidos (marcados obsoletos), a quantidade de blocos livres que forma prematuramente apagados (livres apagados) e a quantidade total de blocos operados, quer por leitura, escrita, sobrescrita ou apagamento (blocos operados).

Tabela 12 – Resultados da simulação dos métodos iniciais

Método	Leituras	Escritas	Apagamentos	Sobrescritas com zeros	Marcados obsoletos	Livres apagados	Blocos operados
Normal	0	0	1271	0	220260	0	81344
Zero-over	0	0	1271	220260	220260	0	301604
Erase	93489	93489	6083	0	0	75563	576290
Sun	4575	4575	4510	21067	21067	64111	318857
Proposto	10387	10387	3889	21701	21701	18712	291371
Huang	93489	93489	6083	220260	0	75563	796550

A partir dos dados da Tabela 12 podem ser observados os seguintes pontos:

- O método normal marca 220.260 blocos como obsoletos, o que indica a quantidade total de blocos ocupados pelos arquivos removidos. Como estes blocos obsoletos precisam ser reciclados antes de poderem ser utilizados novamente, ao longo do processo foi aplicada a reciclagem sobre 1.271 unidades de apagamento. Como cada unidade de apagamento contém 64 blocos, a quantidade total de blocos sobre os quais foram realizadas operações de apagamento foi de 81.344 (64 x 1271).
- O método de sobrescrita com zeros (*zero-overwrite*) foi aplicado a 220.260 blocos, ou seja, a mesma quantidade de blocos que a remoção normal iria marcar como obsoletos. Esta quantidade (220.260) indica a menor quantidade de operações de escrita a serem realizadas para a remoção segura dos arquivos, e é portanto o limite inferior da quantidade de operações necessárias. Como na remoção normal, também foi necessário realizar a reciclagem de 1.271 unidades de apagamento, pois uma sobrescrita com zeros preenche o conteúdo dos blocos com zeros e marca estes blocos como obsoletos, para serem futuramente reciclados. Com a sobrescrita e as reciclagens, o número de blocos operados é de 301.604 (220.260 + 64 x 1.271).
- O método de apagamento (*erase*) envolveu a reciclagem de 6.083 unidades de apagamento e, como existiam blocos válidos (pertencentes a outros arquivos) nestas unidades, foi necessário copiar estes blocos para outras unidades, o que requereu 93.489 operações de leitura e escrita. Observe-se que os apagamentos realizados acabaram reciclando 75.563 blocos livres (*available*), ou seja, reciclaram desnecessariamente blocos que nem sequer haviam sidos alocados a arquivos. O número total de blocos operados foi de 576.290 (93.489 leituras, 93.489 escritas e 6.083 x 64 blocos apagados), o que fornece um limite superior máximo do número de operações necessárias para a remoção dos arquivos.
- O método de Sun, em função do cálculo de sua função de custo, realizou 21.067 sobrescritas com zero, e 4.510 apagamentos de unidades, que por sua vez demandaram 4.575 operações de leitura e escrita para retirar blocos válidos destas unidades antes do seu apagamento. Os apagamentos realizados reciclaram prematuramente 64.111 blocos livres. Com isto, um total de 318.857 operações sobre blocos foram realizadas (4.575 leituras, 4.575 escritas, 21.067 sobrescritas e 4.510 x 64 blocos apagados). Desta forma, o método de Sun apresenta um desempenho intermediário entre o mínimo determinado

pela sobrescrita e o máximo determinado pelo apagamento. Este desempenho é esperado, pois para cada bloco o método de Sun aplica uma função de custo para escolher entre sobrescrita e apagamento.

- O método proposto aplica uma penalidade no cálculo do custo da sobrescrita e no custo do apagamento, com o objetivo de reduzir a quantidade de blocos livres apagados prematuramente. Este objetivo é atingido, pois a quantidade total de blocos livres apagados foi de 18.712, contra 64.111 do método de Sun. Com a penalidade aplicada sobre o custo de um apagamento, a quantidade total de apagamentos foi reduzida de 4.510 (pelo método de Sun) para 3.889, o que explica a redução observada no apagamento de blocos livres, pois a penalidade proposta visa justamente fornecer um custo favorável à sobrescrita (sobre o apagamento) quando existem muitos blocos livres na unidade a ser apagada. Em comparação com o método de Sun, a quantidade de blocos sobrescritos com zero aumentou levemente (21.701 contra 21.067 de Sun), o que se explica pela redução observada na quantidade de apagamentos realizados. Entretanto, a quantidade de leituras e escritas realizadas aumentou significativamente, de 4.575 do método de Sun para 10.387. Isto significa que muito mais blocos válidos foram copiados em função de apagamentos realizados. Mesmo assim, a quantidade total de operações realizadas for de 291.371 (10.387 leituras, 10.387 escritas, 21.701 sobrescritas e 3.889 x 64 blocos apagados), o que é menor que as operações requeridas pelo método de Sun (318.857 operações sobre blocos).
- O método de Huang realiza tanto sobrescrita com zeros como apagamento, e portanto apresenta um resultado que é a soma dos métodos de sobrescrita (220.260 blocos sobrescritos com zeros) e apagamento (6.083 unidades apagadas, com 93.489 cópias de blocos válidos para outras unidades). Por causa disto, é o método que envolve a maior quantidade de blocos operados, com um total de 796.550 (93.489 blocos lidos, 93.489 escritos, 220.260 sobrescritos e 6.083 x 64 blocos apagados).

Apesar do desempenho do método proposto (identificado pela sigla ZOP-ABP no simulador) ser superior ao do método de Sun, principalmente na redução de blocos livres apagados, o grande aumento na quantidade de blocos copiados, ou seja, lidos de uma unidade a ser apagada e escritos em outra unidade disponível, levou à necessidade de uma análise mais completa dos fatores de benefício e de penalidade utilizados no cálculo do custo das operações de sobrescrita e de apagamento.

Assim, o simulador foi ampliado para incluir mais métodos, para permitir análise de outras funções de custo. Isto é discutido nas próximas seções.

7.2 **Refinamento do Método Proposto**

O método proposto originalmente incluía um fator de Penalidade, tanto para as operações de sobrescrita com zeros como para as operações de apagamento, conforme analisado na Seção 5.4.8. Embora o resultado da simulação deste método tenha apresentado vantagens sobre o método de Sun, conforme observado na seção anterior (7.1), ainda existiam pontos a considerar,

como o grande número de operações de escrita e leitura, assim como uma quantidade ainda considerável de blocos livres que foram apagados.

Como ambas as penalidades forneciam um fator que era adicionado ao custo das duas operações, decidiu-se investigar o efeito de não utilizar a penalidade para a operação de sobrescrita com zeros, e somente adicionar a penalidade para a operação de apagamento, pois esta operação é a responsável tanto pela quantidade de cópias de blocos (blocos válidos precisam ser copiados para outra unidade antes do apagamento) como pelo apagamento de blocos livres. Assim, foi implementado no simulador um segundo método proposto, somente com o fator de penalidade no custo da operação de apagamento.

A Tabela 13 mostra os resultados obtidos com esta nova proposta, para a mesma série de experimentos realizados anteriormente, com a remoção total dos mesmos 2437 arquivos. O método proposto originalmente é identificado pela sigla ZOP-ABP, onde a letra P indica a Penalidade tanto na sobrescrita com zeros (ZO) como no apagamento (A). O Segundo método é identificado pela sigla ZO-ABP, para indicar que o custo de uma sobrescrita não é penalizado, enquanto o custo de um apagamento é calculado com o benefício sugerido por Sun (letra B) e com a penalidade proposta (letra P).

Tabela 13 – Resultados da simulação do refinamento do método proposto

Método	Leituras	Escritas	Apagamentos	Sobrescritas com zeros	Marcados obsoletos	Livres apagados	Blocos operados
Normal	0	0	1271	0	220260	0	81344
Zero-over	0	0	1271	220260	220260	0	301604
Erase	93489	93489	6083	0	0	75563	576290
Sun	4575	4575	4510	21067	21067	64111	318857
ZOP-ABP	10387	10387	3889	21701	21701	18712	291371
ZO-ABP	4541	4541	3575	34380	34380	4449	272262
Huang	93489	93489	6083	220260	0	75563	796550

Como pode ser verificado na Tabela 13, uma análise comparativa do método de Sun, o primeiro método proposto (sigla ZOP-ABP) e o segundo método proposto (sigla ZO-ABP) mostra os seguintes resultados:

- O método ZO-ABP apresenta praticamente o mesmo número de cópias de blocos (operações de leitura e escrita) que o método de Sun (4.541 cópias contra 4.575), e resolve portanto o grande número de cópias do método inicialmente proposto, ZOP-ABP (10.387 cópias).
- O método ZO-ABP realizou 3.575 apagamentos, o que é uma redução significativa em relação ao método de Sun (4.510 apagamentos) e também uma redução em relação ao método ZOP-ABP.
- O método ZO-ABP foi, entre os três métodos, o que realizou a menor quantidade de apagamento de blocos livres. Foram apagados somente 4.449 blocos livres, contra 64.111 blocos apagados pelo método de Sun, e 18.712 blocos apagados pelo método ZOP-ABP.

- O método ZO-ABP, entretanto, apresentou um aumento considerável das operações de sobrescrita, realizando 34.380 sobrescritas com zeros, contra 21.067 sobrescritas do método de Sun e 21.701 do método ZOP-ABP. Este aumento era esperado, pois a retirada do fator de penalidade da operação de sobrescrita faz com que o seu custo fique potencialmente menor do que o de uma operação de apagamento, e sobrescritas sejam realizadas mais vezes. Como este aumento é uma consequência da redução observada nas operações de apagamento, o resultado geral pode ser considerado benéfico, pois apagamentos implicam no desgaste da mídia e redução da sua vida útil.
- O método ZO-ABP também foi o que realizou a menor quantidade de operações sobre blocos. Foram 272.262 blocos operados, contra 318.857 blocos do método de Sun e 291.371 blocos do método ZOP-ABP. Como o limite inferior de blocos operados é de 220.260, definido pelo método de sobrescrita (Zero-over, segunda linha da tabela, coluna Sobrescritas), o método ZO-ABP foi dentre os três o que mais se aproximou deste limite.

Por estes resultados, o novo método proposto ZO-ABP foi o que apresentou os melhores resultados. Sua função custo para uma operação de sobrescrita é dada por:

$$Custo_{zo}(i) = N_{blocos\_a\_remover}(i) * E_t$$

e o custo de uma operação de apagamento inclui o Benefício proposto por Sun (blocos apagados não precisam ser sobrescritos) e a Penalidade proposta (apagar blocos livres é desperdício e aumenta o desgaste), sendo dada por:

$$Custo_A(i) = N_{blocos\_válidos}(i) * ((L_t + E_t) + A_t/N_{blocos\_unidade}) + N_{blocos\_livres} * E_t$$

Para cada unidade de apagamento *i* envolvida na remoção segura de um arquivo calculam-se os dois custos e realiza-se a operação associada ao menor custo.

7.3 **Análise Detalhada da Função Custo**

Em função da análise realizada nas seções anteriores (7.1 e 7.2), decidiu-se generalizar as funções de custo das operações de sobrescrita com zeros e de apagamento e aplicar todas as combinações possíveis. Como visto na Seção 5.4.4, o custo de uma sobrescrita é dado por:

$$Custo_{ZO} = T_{zo} = N_{blocos\_a\_remover} * E_t$$

e o custo de um apagamento é expresso pelo tempo necessário para copiar os blocos ainda válidos para outra unidade de apagamento e a seguir apagar a unidade:

$$CustoA = T_A = N_{blocos\_válidos} * (L_t + E_t) + A_t$$

O método de Sun considera ainda um benefício associado à operação de apagamento, pois cria blocos livres que podem ser usados futuramente. Já uma operação de sobrescrita não cria nenhum bloco livre, então seu benefício futuro é zero:

$$Benefício_{zo} = 0$$

$$Benefício_A = ((N_{blocos\_unidade} - N_{blocos\_válidos}(i)) / N_{blocos\_unidade}) * A_t$$

Na análise realizada na Seção 5.4.8, foram propostos dois fatores de penalidade. Na sobrescrita, os blocos que são sobrescritos vão precisar passar por uma operação de apagamento no futuro, antes de poderem ser usados de novo. Para refletir isto, o fator de penalidade é proporcional ao número de blocos sobrescritos:

$$Penalidade_{zo} = (N_{blocos\_a\_remover}(i) / N_{blocos\_unidade}) * A_t$$

No apagamento, por considerar que blocos livres serem apagados antes de serem escritos é um desperdício e acelera o desgaste da memória *flash*, introduz-se um fator de penalidade, proporcional ao número de blocos livres:

$$Penalidade_A(i) = N_{blocos\_livres} * E_t$$

Este fator indica que se perdeu a oportunidade de fazer N_{blocos_livres} escritas.

Considerando todos estes fatores, uma operação de sobrescrita pode ser considerada individualmente (ZO) ou com sua penalidade associada (ZOP). Analogamente, um apagamento pode ser considerado isolado (A), com benefício associado (AB), com penalidade associada (AP) ou com benefício e penalidade (ABP).

Simulando todas as variações possíveis para a função custo, tanto da sobrescrita como do apagamento, obteve-se os resultados observados na Tabela 14.

Tabela 14 – Resultados da simulação de todas as variações da função custo

Método	Leituras	Escritas	Apagamentos	Sobrescritas com zeros	Marcados obsoletos	Livres apagados	Blocos operados
Normal	0	0	1271	0	220260	0	81344
Zero-over	0	0	1271	220260	220260	0	301604
Erase	93489	93489	6083	0	0	75563	576290
ZO-A	23	23	1670	124927	124927	33	231853
ZO-AB (Sun)	4575	4575	4510	21067	21067	64111	318857
ZOP-ABP	10387	10387	3889	21701	21701	18712	291371
ZO-ABP	4541	4541	3575	34380	34380	4449	272262
ZOP-AP	4530	4530	3083	49516	49516	2651	255888
ZO-AP	23	23	1670	124927	124927	33	231853
ZOP-AB	10537	10537	4709	13339	13339	70813	335789
ZOP-A	4530	4530	3415	37602	37602	10866	265222
Huang	93489	93489	6083	220260	0	75563	796550

O método original de Sun é identificado por ZO-AB, o método proposto na Seção 5.4.8 é identificado pela sigla ZOP-ABP e o método refinado em função da análise da Seção 7.2 é identificado por ZO-ABP. Para as demais combinações podem ser observados os seguintes pontos:

- O método ZO-A não considera nem benefício nem penalidades, e é o resultado da comparação simples das operações de sobrescrita e apagamento. Como um apagamento afeta todos os blocos de uma unidade de apagamento, ele somente apresenta vantagem

quando a unidade contém muitos blocos a remover, como pode ser visto na Figura 21 da Seção 7.1. Devido o isto, o método realiza uma quantidade pequena de apagamentos (1.670) e uma quantidade substancial de sobrescritas (124.927, a maior entre todas as combinações analisadas). Isto é uma desvantagem, pois blocos sobrescritos precisam passar por um apagamento posterior antes de serem novamente utilizados.

- O método ZOP-AP aplica penalidades tanto para sobrescrita como para apagamento, sem considerar o fator de benefício de um apagamento. Embora estes fatores favoreçam apagamentos, a quantidade total de apagamentos realizados (3.083) é pequena se comparada com os demais métodos, e a quantidade de sobrescritas (49.516) ainda é grande se comparada com os demais métodos. Assim, apesar do método ZOP-AP ser melhor que o ZO-A, ele ainda tem desempenho inferior em relação aos métodos propostos (o ZO-AB de Sun e o ZO-ABP).
- O método ZO-AP aplica penalidade somente para o apagamento, e com isto o seu resultado é equivalente ao do método ZO-A. Se no método ZO-A um apagamento já era aplicado poucas vezes, a adição de uma penalidade não aumenta as chances de sua aplicação. Nas simulações realizadas o seu desempenho foi inclusive idêntico ao do método ZO-A.
- O método ZOP-AB aplica penalidade na sobrescrita e benefício no apagamento. Com isto a quantidade de apagamentos (4.709) é bem significativa e é a maior entre todas as combinações realizadas. Como consequência, a quantidade de sobrescritas (13.339) é bem modesta, e é a menor entre todas as combinações. Entretanto, o método ZOP-AB gera uma grande quantidade de cópias de blocos (10.537) e faz com que uma grande quantidade de blocos livres (70.813) seja apagada prematuramente.
- O método ZOP-A aplica penalidade somente para a sobrescrita, sem entretanto considerar benefício para o apagamento. Embora tanto a quantidade de apagamentos (3.415) como a de sobrescritas (37.602) sejam razoáveis, a grande quantidade de blocos livres apagados (10.866) compromete o desempenho deste método. Em consequência, apesar do ZOP-A ter desempenho melhor que o ZOP-ABP, ele ainda é inferior ao método ZO-ABP.

Analisando os diversos métodos em função da quantidade das operações realizadas (apagamentos, sobrescritas, blocos livres apagados, blocos operados e blocos copiados), obtém-se a Tabela 15. À direita de cada coluna que indica a quantidade de operações realizadas encontra-se a classificação dos métodos em ordem crescente. Um método de bom desempenho deve ser balanceado em relação às operações de apagamento e sobrescrita, pois uma quantidade muito grande de uma destas operações e muito pequena da outra operação indica a existência de deficiências no método. Realizar muitas sobrescritas e poucos apagamentos significa simplesmente adiar apagamentos que devem ser realizados para reaproveitar os blocos, e realizar poucas sobrescritas e muitos apagamentos significa acelerar o desgaste da mídia. Desta forma, na Tabela 15 podem-se descartar os métodos que estejam classificados entre os três primeiros de uma das operações (apagamento ou sobrescrita) e simultaneamente entre os três últimos da outra operação (sobrescrita ou apagamento). Nesta condição estão os métodos de Sobrescrita (classificação 1 e 10), de Apagamento (classificação 10 e 1), o método ZO-AP (classificação 3 e 9), o método ZOP-AB (classificação 9 e 2) e o método ZO-A (classificação 2 e 8). O método de Huang

também pode ser descartado, por realizar as duas operações e portanto apresentar as duas desvantagens associadas.

Tabela 15 – Classificação dos métodos pela quantidade de apagamentos e sobrescritas

Método	Apagamentos	class.	Sobrescritas	class.
Zero-over	1271	1	220260	10
Erase	6083	10	0	1
ZO-A	1670	2	124927	8
ZO-AB (Sun)	4510	8	21067	3
ZOP-ABP	3889	7	21701	4
ZO-ABP	3575	6	34380	5
ZOP-AP	3083	4	49516	7
ZO-AP	1670	3	124927	9
ZOP-AB	4709	9	13339	2
ZOP-A	3415	5	37602	6
Huang	6083	11	220260	11

Para refinar a classificação dos métodos, eles podem ser analisados pela quantidade de blocos livres apagados, pela quantidade total de blocos operados e pela quantidade de cópias de blocos realizadas. O resultado pode ser visto na Tabela 16. Um critério importante é a quantidade de blocos livres (ainda não utilizados para uma escrita) que são apagados prematuramente, e neste sentido os métodos ZO-AB, ZOP-ABP e ZOP-A apresentam quantidades bem mais significativas que os métodos ZO-ABP e ZOP-AP.

Tabela 16 – Classificação dos métodos em função da quantidade de operações

Método	Apagamentos		Sobrescritas		Livres Apagados		Blocos Operados		Blocos Copiados	
ZO-AB (Sun)	4510	5	21067	1	64111	5	318857	5	4575	4
ZOP-ABP	3889	4	21701	2	18712	4	291371	4	10387	5
ZO-ABP	3575	3	34380	3	4449	2	272262	3	4541	3
ZOP-AP	3083	1	49516	5	2651	1	255888	1	4530	1/2
ZOP-A	3415	2	37602	4	10866	3	265222	2	4530	1/2

Pela Tabela 16, os métodos que podem ser considerados de bom desempenho (balanceamento entre apagamentos e sobrescritas e baixa quantidade de blocos livres apagados) são o ZO-ABP (analisado na Seção 7.2) e o ZOP-AP. Entretanto, o método ZO-ABP é mais balanceado entre apagamento e sobrescritas.

7.4 **Análise da influência do tamanho da unidade de apagamento**

Memórias *flash* atualmente têm unidades de apagamento que agrupam 32, 64 ou 128 blocos (ver Seção 6.6), sendo que 32 blocos por unidade são encontrados somente em memórias de pequena

capacidade (abaixo de 128 MBytes). Para avaliar a influência do tamanho da unidade de apagamento no desempenho dos diversos métodos, foram simuladas as mesmas operações de remoção de arquivos para tamanhos de 32, 64 e 128 blocos por unidade de apagamento. Os resultados são analisados a seguir, para os métodos da Tabela 16. Resultados mais detalhados podem ser encontrados no Apêndice A.

A Tabela 17 mostra a quantidade de apagamentos realizados por cada método em função do tamanho, em blocos, da unidade de apagamento. Como pode ser observado, a quantidade de apagamentos realizada diminui conforme o tamanho da unidade aumenta. Este resultado condiz com as expectativas, pois se a unidade de apagamento contém mais blocos, os arquivos de mesmo tamanho tendem a ocupar menos unidades de apagamento.

Tabela 17 – Quantidade de apagamentos por tamanho de unidade

Método	32	64	128
ZO-AB (Sun)	7713	4510	2806
ZOP-ABP	7386	3889	2282
ZO-ABP	7025	3575	2032
ZOP-AP	6076	3083	1719
ZOP-A	6312	3415	2044

Plotando as quantidades de apagamentos da Tabela 17, obtém-se o gráfico da Figura 27. Como pode ser observado, todos os cinco métodos apresentam curvas semelhantes, o que permite deduzir que todos são afetados da mesma maneira pelo tamanho da unidade de apagamento.

Figura 27 – Quantidade de apagamentos por tamanho de unidade

A Tabela 18 mostra a quantidade de sobrescritas realizadas por cada um dos cinco métodos em função do tamanho, em blocos, da unidade de apagamento. Como pode ser observado, a quantidade de blocos sobrescritos com zeros aumenta conforme o tamanho da unidade também

aumenta. Este resultado comprova que os cinco métodos escolhidos garantem um bom balanceamento entre apagamentos e sobrescritas, pois a redução das operações de apagamento implica em um aumento correspondente nas operações de sobrescrita.

Tabela 18 – Quantidade de sobrescritas por tamanho de unidade

Método	32	64	128
ZO-AB (Sun)	13084	21067	37137
ZOP-ABP	9814	21701	39076
ZO-ABP	17916	34380	55143
ZOP-AP	34463	49516	74555
ZOP-A	29258	37602	52297

A Figura 28 mostra os dados da Tabela 18 representados em forma de um gráfico. Embora as posições relativas de cada método mudem para diferentes tamanhos de unidades de apagamento, todos os métodos apresentam o mesmo comportamento conforme o tamanho da unidade de apagamento aumenta.

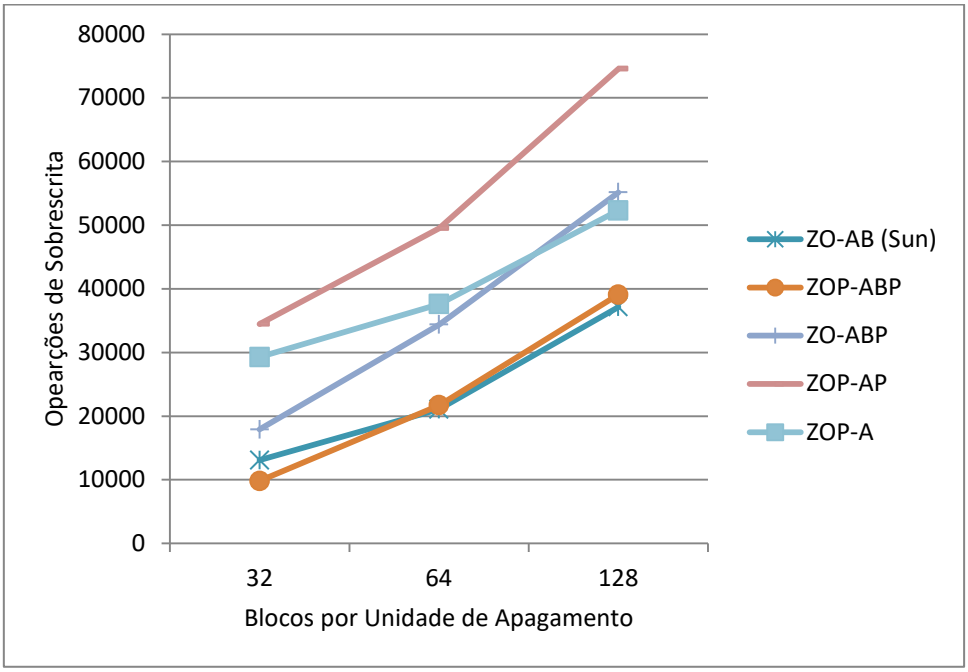


Figura 28 – Quantidade de sobrescritas por tamanho de unidade

Embora a quantidade de apagamentos e sobrescritas realizados, analisados nas tabelas e figuras acima, seja o melhor parâmetro para avaliar os métodos de remoção, também é importante que um método eficiente não apague uma quantidade significativa de blocos livres quando estiver reciclando uma unidade de apagamento. Neste sentido, a Tabela 19 mostra a quantidade de blocos livres que foram apagados antes de serem utilizados por um arquivo. Como pode ser observado, o método de Sun é o que desperdiça a maior quantidade de blocos livres. Conforme foi analisado na Seção 5.4.8, esta característica era esperada, pois o método não considera a existência de blocos livres ao calcular o custo de cada operação. Já o fator de Penalidade introduzido nos métodos propostos (Seção 7.1 e Seção 7.2) considera uma desvantagem apagar

blocos livres, e em consequência os métodos “ABP” e “AP” apresentam desempenho melhor neste aspecto.

Tabela 19 – Quantidade de blocos livres apagados por tamanho de unidade

Método	32	64	128
ZO-AB (Sun)	25561	64111	127419
ZOP-ABP	10961	18712	52321
ZO-ABP	3623	4449	28656
ZOP-AP	847	2651	5830
ZOP-A	3261	10866	45147

A Figura 29 mostra os dados da Tabela 19 sob forma gráfica. Embora os cinco métodos apresentem o mesmo comportamento, ou seja, aumento da quantidade de blocos livres apagados conforme o tamanho da unidade de apagamento cresce, pode-se verificar que o método de Sun apresenta uma quantidade significativamente maior de blocos livres apagados.

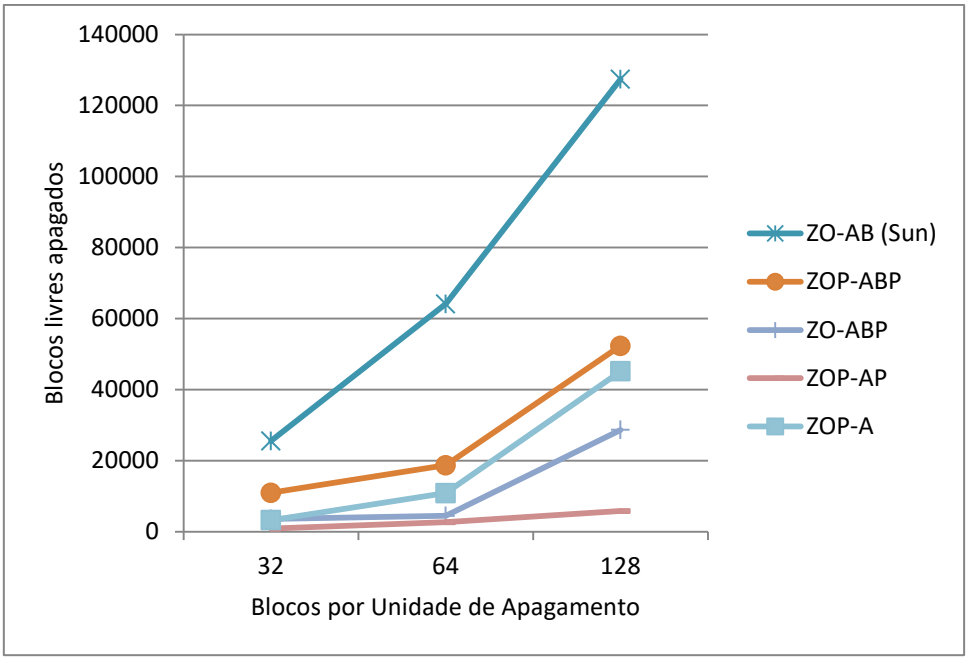


Figura 29 – Quantidade de blocos livres apagados por tamanho de unidade

A Tabela 20 mostra a quantidade total de setores operados (apagados, sobrescritos ou copiados) em cada um dos métodos. Como pode ser verificado, esta quantidade aumenta em função do tamanho da unidade de apagamento.

Tabela 20 – Quantidade de blocos operados por tamanho de unidade

Método	32	64	128
ZO-AB (Sun)	262264	318857	420355
ZOP-ABP	256706	291371	372022
ZO-ABP	245080	272262	339467
ZOP-AP	231255	255888	317149
ZOP-A	233602	265222	336549

Plotando os dados da Tabela 20 em um gráfico, obtém-se o resultado mostrado na Figura 30. Todos os cinco métodos variam da mesma forma, ou seja, são afetados da mesma forma conforme o tamanho da unidade de apagamento aumenta. O método de Sun tem o maior aumento, pelo fato de apagar uma quantidade significativa de setores livres.

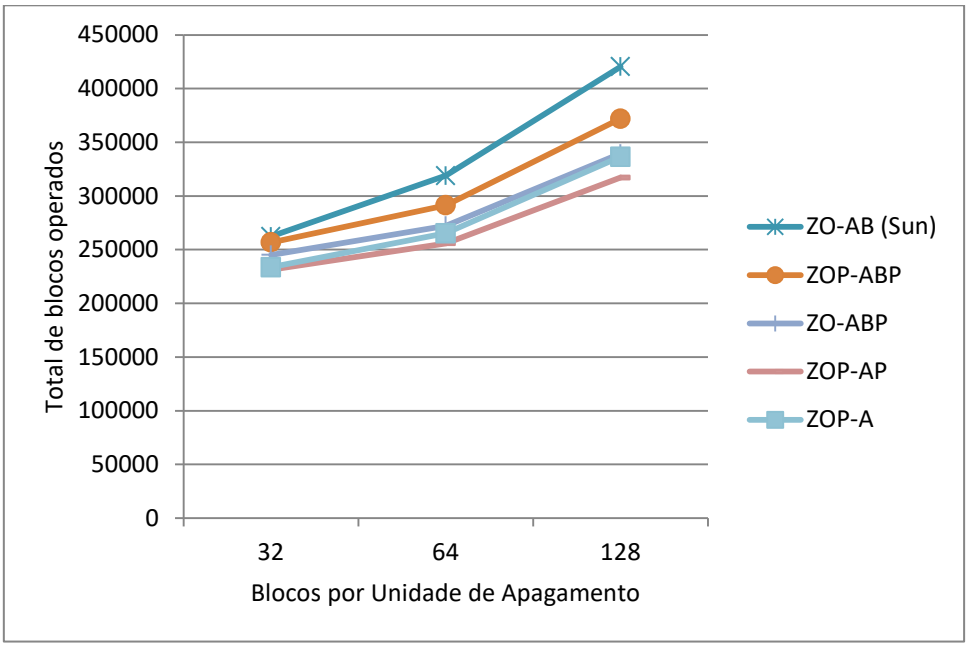


Figura 30 – Quantidade de blocos operados por tamanho de unidade

A Tabela 21 mostra a quantidade de blocos válidos (de outros arquivos) copiados de uma unidade de apagamento que será apagada (para outra unidade) quando se realiza a remoção de um arquivo. Esta quantidade aumenta conforme o tamanho da unidade cresce, o que é explicado pelo fator de uma unidade de tamanho maior tem mais probabilidade de conter blocos de vários arquivos distintos.

Tabela 21 – Quantidade de blocos copiados por tamanho de unidade

Método	32	64	128
ZO-AB (Sun)	1182	4575	12025
ZOP-ABP	5270	10387	20425
ZO-ABP	1182	4541	12114
ZOP-AP	1180	4530	11281
ZOP-A	1180	4530	11310

A quantidade de blocos copiados da Tabela 21 é ilustrada sob forma gráfica na Figura 31. Observe que os métodos têm praticamente o mesmo comportamento, exceto o método originalmente proposto na Seção 7.1 (ZOP-ABP). Entretanto, este método foi aprimorado na Seção 7.2, dando origem ao método ZO-ABP, que não apresenta o crescimento desproporcional do método ZOP-ABP.

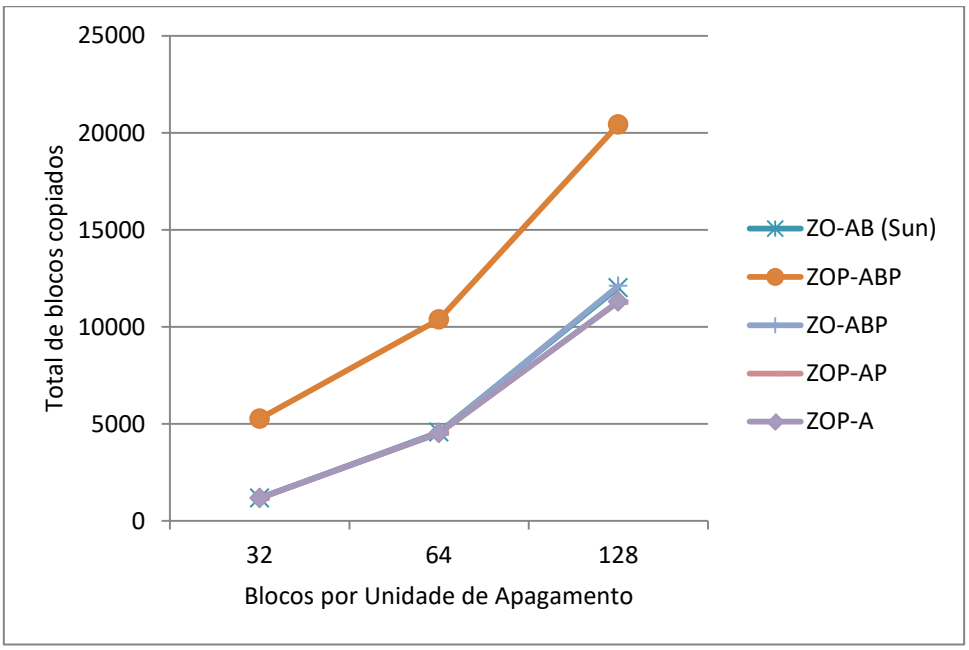


Figura 31 – Quantidade de blocos copiados por tamanho de unidade

Em conclusão, todos os cinco métodos são dependentes do tamanho da unidade de apagamento, mas todos sofrem sempre o mesmo tipo de influência nas medidas realizadas (quantidade de apagamentos, quantidade de sobrescritas, quantidade de blocos livres apagados, quantidade total de blocos operados e quantidade de blocos copiados).

7.5 **Análise da influência do tempo das operações**

Os cincos métodos escolhidos na Seção 7.3 também devem ser analisados em função dos tempos das operações de uma memória *flash*, ou seja, os tempos de leitura de um bloco, de escrita de um bloco e de apagamento de uma unidade. Para realizar esta análise, as remoções de arquivos foram simuladas para cinco características distintas das memórias *flash* (Seção 6.6):

- Leitura de 25 µs, escrita de 220 µs e apagamento de 500 µs (25-220-500)
- Leitura de 25 µs, escrita de 220 µs e apagamento de 1500 µs (25-220-1500)
- Leitura de 25 µs, escrita de 300 µs e apagamento de 2000 µs (25-220-500)
- Leitura de 10 µs, escrita de 50 µs e apagamento de 3000 µs (10-50-3000)
- Leitura de 30 µs, escrita de 250 µs e apagamento de 3000 µs (30-250-3000)

A análise dos dados é realizada considerando-se as mesmas medidas da seção anterior (Seção 7.4), ou seja, quantidade de operações de apagamentos, quantidade de operações de sobrescrita, quantidade de blocos livres apagados, quantidade total de blocos operados e quantidade de blocos copiados.

A Tabela 22 mostra a quantidade de operações de apagamentos realizadas por cada um dos cinco métodos. Embora não sejam notadas variações significativas em cada um dos métodos, alguns destes métodos são mais estáveis, ou seja, apresentam menor variação em função da mudança das características da memória *flash*, como pode ser visto na Figura 32.

Tabela 22 – Quantidade de apagamentos por tempos de operações

	25-220-500	25-220-1500	25-300-2000	10-50-3000	30-250-3000
ZO-AB (Sun)	4693	4669	4692	4510	4645
ZOP-ABP	3681	3794	3794	3889	3805
ZO-ABP	3662	3651	3662	3575	3640
ZOP-AP	3654	3611	3626	3083	3585
ZOP-A	4656	4208	4231	3415	3988

Analisando o comportamento dos cinco métodos na Figura 32, pode-se verificar que o método ZOP-A é o que apresenta maior variação em função da mudança dos tempos das operações. Já o método ZOP-AB (de Sun) e o método ZO-ABP (proposto na Seção 7.2) são os que permanecem mais constantes em função da variação dos tempos das operações.

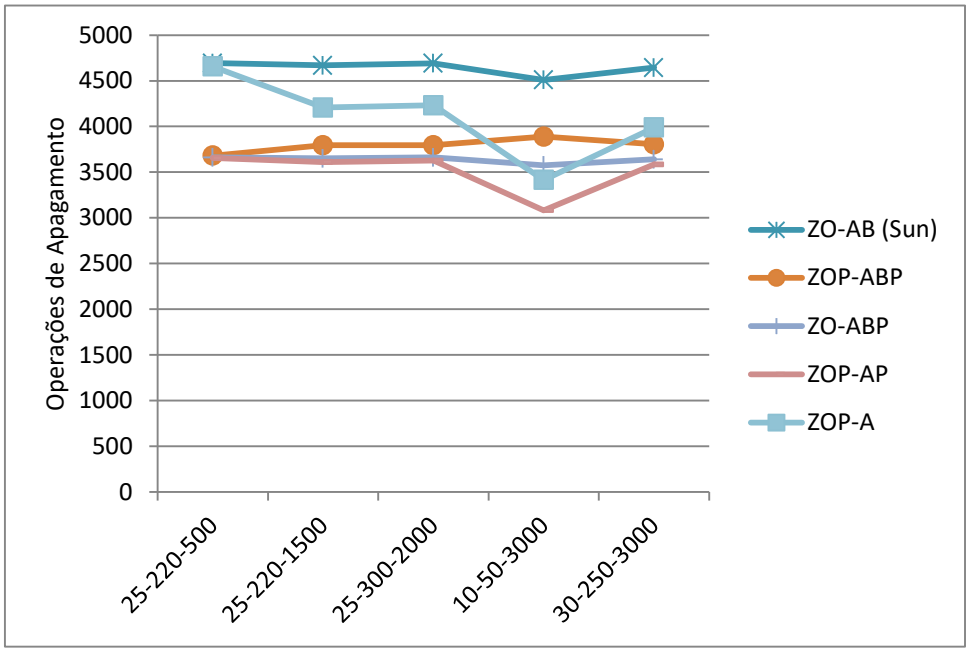


Figura 32 – Quantidade de apagamentos por tempos de operações

Na Tabela 23 são mostradas as quantidades de operações de sobrescrita de blocos realizadas por cada um dos cinco métodos. Em contraste com o número de operações de apagamento (Tabela 22), agora existem variações significativas em alguns métodos, como é ilustrado no gráfico da Figura 33.

Tabela 23 – Quantidade de sobrescritas por tempos de operações

	25-220-500	25-220-1500	25-300-2000	10-50-3000	30-250-3000
ZO-AB (Sun)	13934	14621	13941	21067	15421
ZOP-ABP	28606	25137	25137	21701	24553
ZO-ABP	29273	29753	29273	34380	30336
ZOP-AP	29536	30632	30240	49516	31552
ZOP-A	14002	16981	16281	37602	19513

Pelo comportamento de cada método na Figura 33, pode-se concluir que os métodos ZOP-AP e ZOP-A são os que apresentam maiores variações. Já o método ZO-ABP (proposto na Seção 7.2) é o que tem menor variação em função dos tempos das operações.

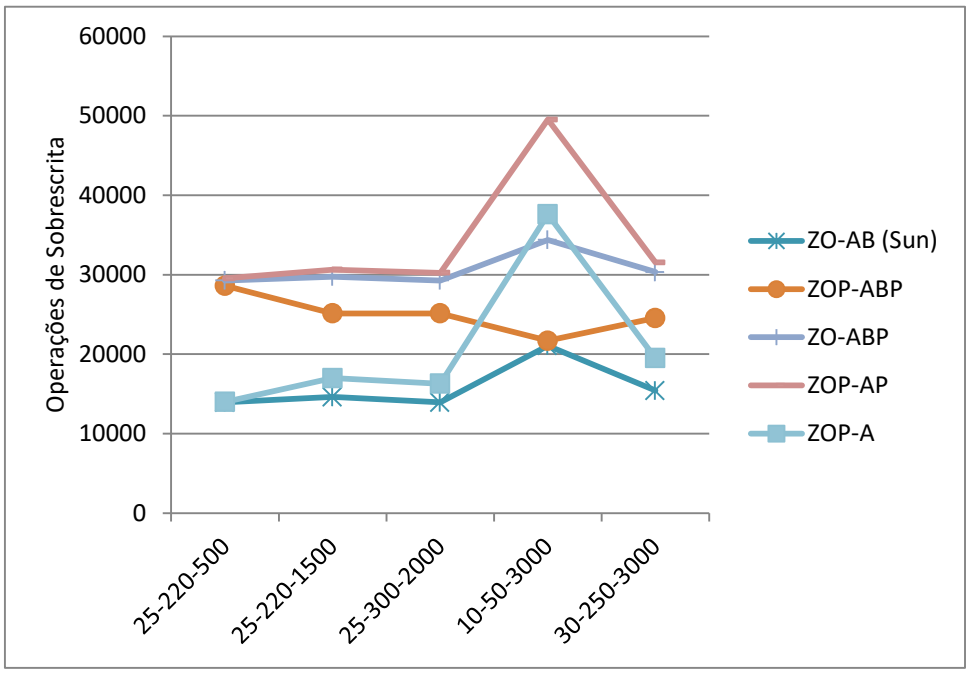


Figura 33 – Quantidade de sobrescritas por tempos de operações

Em relação à quantidade de blocos livres apagados prematuramente pelo apagamento da unidade em que se encontram, a Tabela 24 mostra como se comportam cada um dos cinco métodos, e a Figura 34 ilustra o gráfico correspondente.

Tabela 24 – Quantidade de blocos livres apagados por tempos de operações

	25-220-500	25-220-1500	25-300-2000	10-50-3000	30-250-3000
ZO-AB (Sun)	70459	69561	70408	64111	68706
ZOP-ABP	5493	12645	12645	18712	13331
ZO-ABP	4998	4901	4998	4449	4891
ZOP-AP	4693	4539	4167	2651	3978
ZOP-A	68110	41660	42459	10866	28585

Conforme as curvas da Figura 34, o método ZOP-A apresenta grande dependência dos tempos das operações. O método ZO-AB (de Sun), não sofre muita variação, mas, conforme já constatado na Tabela 19, é o que mais desperdiça blocos livres, justamente por não considerar este tipo de bloco nas suas funções de custo. Os métodos ZO-ABP e ZOP-AP são os mais independentes dos tempos das operações, além de apresentarem as menores quantidades de blocos livres que são apagados.

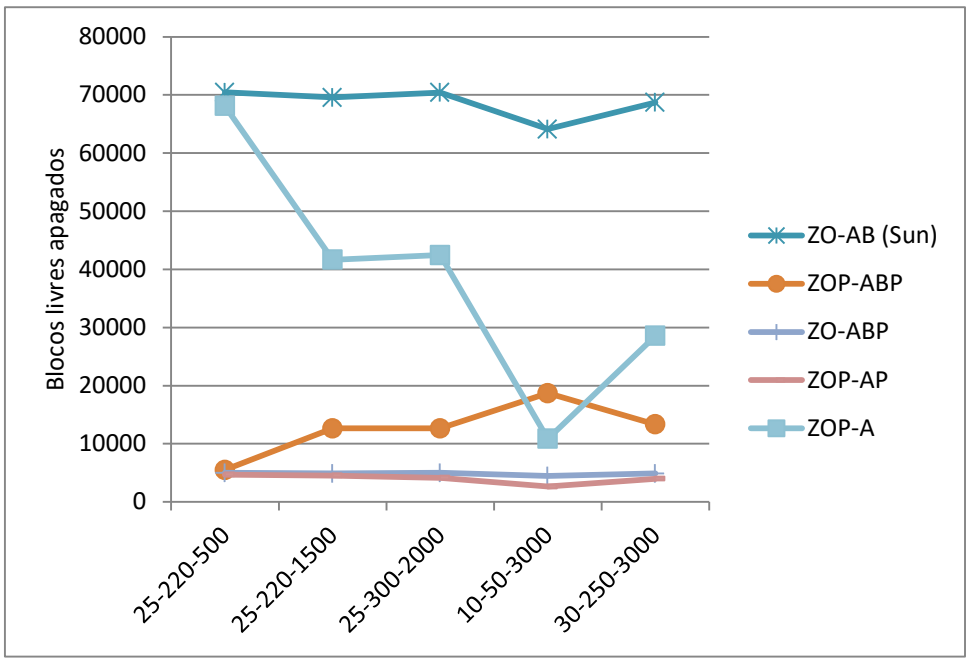


Figura 34 – Quantidade de blocos livres apagados por tempos de operações

A Tabela 25 apresenta a quantidade total de blocos operados, quer por leitura e escrita (cópia), sobrescrita (com zeros) ou por apagamento. Não se observam variações significativas em função dos tempos das operações.

Tabela 25 – Quantidade de blocos operados por tempos de operações

	25-220-500	25-220-1500	25-300-2000	10-50-3000	30-250-3000
ZO-AB (Sun)	334020	331909	333951	318857	329811
ZOP-ABP	284790	288651	288651	291371	288801
ZO-ABP	282933	281433	282933	272262	279936
ZOP-AP	282534	279604	281432	255888	277402
ZOP-A	331696	304551	306525	265222	291329

A Figura 35 ilustra os dados da Tabela 25 sob forma gráfica. Os métodos ZOP-A e ZOP-AP apresentam alguma variação quando os tempos das operações mudam, enquanto que os demais (ZO-AB, ZOP-ABP e ZO-ABP) mostram pouca variação.

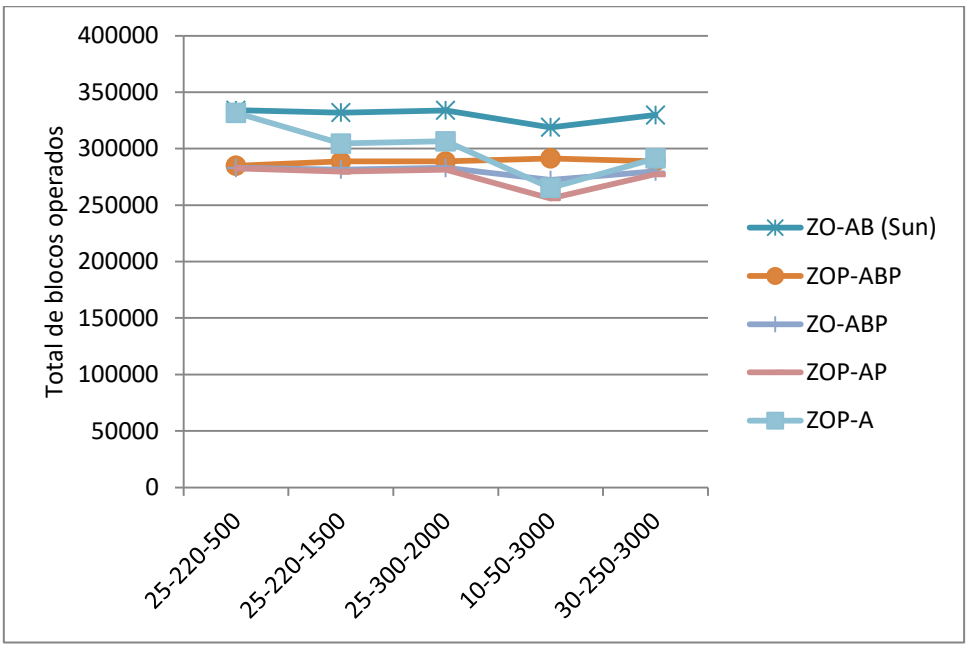


Figura 35 – Quantidade de blocos operados por tempos de operações

A Tabela 26 apresenta a quantidade de blocos copiados de uma unidade de apagamento para outra, quando a primeira deve ser apagada. Com exceção do método ZOP-ABP, todos os demais apresentam variações, como ilustrado na Figura 36.

Tabela 26 – Quantidade de blocos copiados por tempos de operações

	25-220-500	25-220-1500	25-300-2000	10-50-3000	30-250-3000
ZO-AB (Sun)	9867	9236	9861	4575	8555
ZOP-ABP	10300	10349	10349	10387	10364
ZO-ABP	9646	9008	9646	4541	8320
ZOP-AP	9571	8934	9564	4530	8205
ZOP-A	9855	9129	9730	4530	8292

Entretanto, as variações observadas não são relevantes, pois a quantidade de blocos copiados em cada um dos métodos não é um parâmetro significativo no desempenho destes métodos. Como analisado na Seção 7.3, os fatores mais importantes são um balanceamento entre apagamentos e sobrescritas e um número reduzido de blocos livres prematuramente apagados.

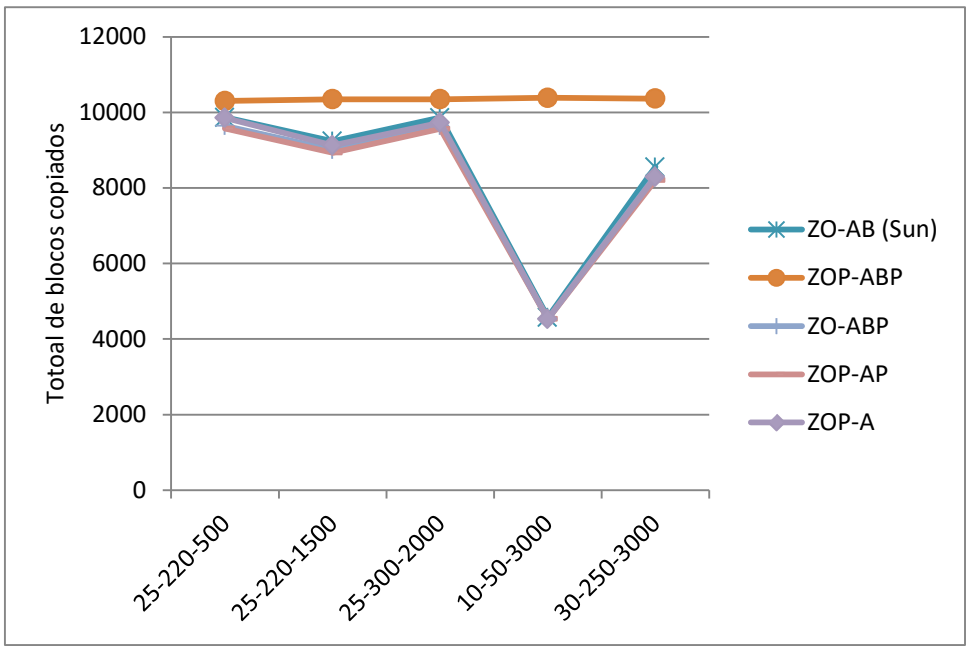


Figura 36 – Quantidade de blocos copiados por tempos de operações

Em função dos dados observados da Tabela 22 até a Tabela 26 (Figura 32 a Figura 36), pode-se concluir que os métodos ZOP-A e ZOP-AP são fortemente dependentes dos tempos das operações de leitura, escrita e apagamento. Tanto o método ZO-AB (proposto por Sun) e o método ZOP-ABP, originalmente proposto (Seção 7.1), não apresentam uma dependência tão acentuada. O método ZO-ABP, proposto em função do refinamento efetuado na Seção 7.2, é o que apresenta menor dependência dos tempos das operações.

8 ANÁLISE DOS EXPERIMENTOS

No capítulo anterior os métodos de remoção por sobrescrita e apagamento foram analisados sob aspectos quantitativos. Neste capítulo é realizada uma análise focada mais em aspectos qualitativos, como vantagens e desvantagens de cada método, assim como suas resistências a ameaças, ou seja, a recuperação de arquivos removidos.

8.1 **Análise qualitativa dos métodos de remoção**

Os métodos utilizados para remover arquivos podem ser classificados em três grandes grupos: remoção normal, remoção por sobrescrita e apagamento e remoção por apagamento criptográfico. Enquanto a emoção normal, ou seja, a remoção realizada pelo sistema operacional quando um arquivo é deletado, permite a recuperação do conteúdo do arquivo, os dois outros métodos tem como objetivo realizar uma remoção segura, que impeça uma posterior recuperação.

8.1.1 Método Normal

O método normal, utilizado pelo sistema operacional quando deve remover um arquivo, simplesmente remove a entrada do arquivo no diretório e libera os blocos que estavam sendo utilizados pelo arquivo, sem entretanto apagar o conteúdo destes blocos.

- **Vantagens**: extremamente simples e rápido. Para liberar um bloco físico, a FTL simplesmente marca este bloco como inválido (obsoleto). Na próxima reciclagem (*garbage collection*), os blocos obsoletos são potenciais candidatos para serem apagados.
- **Desvantagens**: o conteúdo do arquivo não é removido, e pode ser recuperado. Assim, o método normal não cumpre o objetivo principal deste trabalho, que é a remoção segura. Além disso, um bloco obsoleto não pode ser reutilizado até que seja reciclado.
- **Uso recomendado**: não recomendado quando se deseja remoção segura de arquivos com conteúdo sensível.

8.1.2 Métodos de Sobrescrita e Apagamento

Nesta categoria estão todos os métodos que eliminam o conteúdo do arquivo através de operações de sobrescrita com zeros ou de apagamento. O trabalho desenvolvido concentrou-se nestes métodos, que foram estudados na Seção 5.4, foram implementados no Simulador e foram analisados em termos práticos nos experimentos do Capítulo 7.

O método de Sobrescrita pura (denominado de Zero-over no Simulador) somente usa operações de sobrescritas com zeros para eliminar o conteúdo dos blocos físicos. Um bloco sobrescrito é marcado como obsoleto, para ser apagado, e volta a ser utilizado quando da próxima reciclagem.

- **Vantagens**: sobrescrever com zeros e marcar como obsoleto é uma operação simples que pode ser implementada de forma direta pela FTL, sem necessitar de cálculo de funções de custo.

- **Desvantagens**: blocos sobrescritos precisam ser apagados antes de poderem ser reutilizados, então o uso exclusivo de operações de sobrescrita não elimina a necessidade de realizar operações de apagamento posteriormente. Este caso fica evidente quando devem ser removidos todos os blocos de uma mesma unidade de apagamento. Sobrescrever primeiro os blocos para imediatamente depois apagar toda a unidade implica em duas operações redundantes, e a sobrescrita se torna desnecessária.
- **Uso recomendado**: apesar de realizar remoção segura, não é recomendado por realizar operações desnecessárias, como no caso descrito acima.

O método de Apagamento puro (denominado de Erase no Simulador) somente usa operações de apagamento de unidade para eliminar o conteúdo dos blocos físicos. Como entretanto uma unidade pode conter blocos de outros arquivos, antes do apagamento deve-se copiar estes blocos válidos para outra unidades.

- **Vantagens**: blocos apagados podem imediatamente ser reutilizados. Não é mais necessário realizar reciclagem de blocos obsoletos. Estes blocos não existem mais, pois através de um apagamento os blocos passam diretamente de válidos para livres (*available*).
- **Desvantagens**: realiza uma grande quantidade de cópias de blocos (os blocos de outros arquivos que devem ser movidos para outra unidade antes do apagamento) e também apaga muitos blocos prematuramente (os blocos livres que serão apagados simplesmente por estarem na mesma unidade que vai ser apagada). Também realiza uma grande quantidade de apagamentos (veja-se, por exemplo, a Tabela 14 na Seção 7.3), o que acelera o desgaste do meio físico. Observe-se que apagamentos ocorrem mesmo em situações extremas, onde se deseja remover um único bloco de uma unidade de 128 blocos, por exemplo.
- **Uso recomendado**: apesar de realizar remoção segura, não é recomendado por realizar um grande número de operações, que aumentam o desgaste do meio.

O método híbrido de Sun utiliza uma função de custo para decidir entre operações de sobrescrita ou de apagamento (Seção 5.4.4), de forma a procurar utilizar as vantagens destas operações sem incorrer nas desvantagens.

- **Vantagens**: ao realizar a operação de menor custo, consegue eliminar o conteúdo dos blocos físicos com uma menor quantidade de operações que o método da sobrescrita pura ou o método do apagamento puro (veja-se, por exemplo, a Tabela 14 na Seção 7.3). Ao utilizar uma menor quantidade de apagamentos necessários, também não acelera tanto o desgaste do meio. Além disto, blocos apagados podem ser imediatamente reutilizados.
- **Desvantagens**: blocos sobrescritos precisam passar por uma reciclagem futura antes de poderem ser novamente utilizados, então ainda requer a realização de *garbage collection*. Não considera a existência de blocos livres quando calcula o custo de um apagamento, então pode realizar apagamentos desnecessários. Por exemplo, se uma unidade de 128 blocos contiver 18 blocos a remover e 110 blocos livres, o método de Sun decide pelo apagamento da unidade, quando a sobrescrita dos 18 blocos seria mais racional.

- **Uso recomendado**: apesar das desvantagens, realiza remoção segura e pode ser utilizado. O método proposto neste trabalho, entretanto, tem melhor desempenho, como analisado na Seção 7.2.

O método ZO-ABP, proposto por este trabalho, também utiliza uma função de custo para decidir entre operações de sobrescrita ou apagamento. Diferentemente do método de Sun (ZO-AB), que atribui um fator de Benefício ao apagamento, o método ZO-ABP inclui não só este Benefício, mas também uma Penalidade quando existirem blocos livres na unidade a ser apagada.

- **Vantagens**: apresenta as mesmas vantagens do método de Sun, mas atinge o seu objetivo com uma quantidade menor de operações e um menor desperdício de blocos livres apagados (veja-se, por exemplo, a Tabela 14 na Seção 7.3).
- **Desvantagens**: tende a realizar mais operações de sobrescrita que o método de Sun, então potencialmente estar mais dependente de *garbage collection* para reciclar os blocos sobrescritos.
- **Uso recomendado**: realiza remoção segura de forma mais eficiente que o método de Sun, então é o mais recomendado quando se preservar o meio físico contra desgaste.

O método de Huang segue rigorosamente as recomendações do NIST [40] para remoção segura de arquivos, que especificam uma remoção em dois passos, onde incialmente os blocos são preenchidos com um padrão binário em um passo e escritos com o padrão complementar no passo seguinte. Como a sobrescrita preenche todos os bits com zeros e o apagamento coloca todos os bits em ‘1’, Huang propõe usar uma sobrescrita seguida de um apagamento.

- **Vantagens**: de todos os métodos, é o único que segue todas as recomendações do NIST para remoção do conteúdo original dos blocos físicos da memória *flash*.
- **Desvantagens**: é o método de maior custo, pois realiza duas operações (sobrescrita e apagamento) enquanto os demais métodos escolhem entre uma das duas. Além disto, é o método que causa o maior desgaste do meio físico, devido à grande quantidade de apagamentos realizados.
- **Uso recomendado**: por seguir as recomendações do NIST, recomenda-se utilizar este método quando os arquivos a serem removidos contêm informação altamente sensível (governamentais ou empresariais). Pelo alto grau de desgaste, não é recomendado para arquivos com baixo grau de confidencialidade (pessoais ou domésticos).

8.1.3 Métodos de Apagamento Criptográfico

Os métodos de remoção por sobrescrita e apagamento seguem a mesma metodologia tradicionalmente empregada para meios magnéticos, ou seja, apagar o conteúdo original escrevendo por cima deste conteúdo um padrão binário neutro (que não contenha informação). Como uma alternativa a estes métodos, podem ser utilizados dispositivos de armazenamento com cifragem e decifragem integradas (Seção 4.2). Nestes dispositivos, se o uso de criptografia não puder ser desabilitado, nenhum arquivo pode ser acessado em forma legível, e a confidencialidade da informação é garantida pelo segurança do algoritmo criptográfico utilizado. No contexto do método de Apagamento Criptográfico não se deseja um sistema de arquivo seguro, mas sim

impedir o acesso ao conteúdo original do arquivo através da remoção da chave de criptografia associada a este arquivo. O conteúdo cifrado do arquivo continua podendo ser lido, mas é inútil sem a chave criptográfica.

No Apagamento Criptográfico o problema da remoção segura de um arquivo resume-se à remoção de uma chave de criptografia. Devido a isto, o custo do apagamento criptográfico é constante – remoção de uma chave por arquivo, independente do tamanho ou da localização do arquivo no meio de armazenamento. Devido a esta característica os métodos de apagamento criptográfico não foram incluídos no Simulador.

Na utilização de um método de Apagamento Criptográfico, as normas do NIST impõe uma série de exigências:

- Os métodos criptográficos utilizados devem estar totalmente contidos no dispositivo. Em termos de uma memória *flash*, isto significa que o a algoritmo de cifragem e decifragem deve ser implementado na FTL, o que aumenta consideravelmente a sua complexidade.
- As chaves criptográficas utilizadas devem permanecer sempre no interior do dispositivo, não sendo permitida a existência de cópias externas. Isto significa que a FTL também deve ser responsável pela geração e gerência de chaves.
- Todos os arquivos armazenados no dispositivo devem ser cifrados, não sendo permitido a existência de informação não-cifrada. Isto significa que a cifragem e a decifragem devem ser executadas sem o uso de arquivos intermediários.

Nos Capítulos 4 e 5 foram analisados dois métodos que empregam Apagamento Criptográfico: o método de J. Lee [29] (Seção 5.4.5) e o método de B. Lee [28] (Seção 5.4.7).

O método de J. Lee, quando precisa remover a chave de um arquivo, inicialmente sobrescreve esta chave com zeros, a seguir copia todas as demais chaves que estão na mesma unidade de apagamento para outra unidade e, como último passo, apaga esta unidade.

- **Vantagens**: custo constante por arquivo, pois somente sobrescreve e apaga a chave associada ao arquivo a ser removido. Os blocos do arquivo são marcados simplesmente como obsoletos, para serem futuramente reciclados.
- **Desvantagens**: necessita do algoritmo criptográfico implementado na FTL, que também deve gerar e gerenciar chaves. Assim, não é possível alterar nem o algoritmo de criptografia nem o de geração de chaves, caso alguma fraqueza seja descoberta nestes algoritmos. Toda a leitura/escrita de blocos implica adicionalmente da decifragem/cifragem do conteúdo destes blocos, o que aumenta os tempos de leitura e escrita de arquivos. Blocos cifrados ainda podem ser recuperados e armazenados em um meio externo, para processamento futuro.
- **Uso recomendado**: é uma alternativa classificada pelo NIST entre os métodos de apagamento seguro, mas é complexa de implementar na FTL e necessita de maiores estudos que comprovem sua validade e eficiência.

O método de B. Lee é mais simples, pois somente realiza uma sobrescrita da chave com zeros. Não realiza o apagamento da unidade com o argumento que esta unidade será apagada futuramente, quando todas as suas chaves tiverem sido sobrescritas e a unidade for reciclada.

- **Vantagens**: além de custo constante por arquivo, é o método mais simples, pois somente realiza uma única operação (escrita de zeros na chave) para realizar a remoção de um arquivo.
- **Desvantagens**: apresenta todas as desvantagens do método de J. Lee, e não segue rigorosamente as recomendações do NIST de realizar duas operações (sobrescrita e apagamento). Isto, em termos teóricos, facilitaria a recuperação da chave.
- **Uso recomendado**: necessita de maiores análises práticas antes de poder ser adotado como um método efetivo.

De um modo geral, métodos que utilizam Apagamento Criptográfico apresentam baixo custo na remoção de um arquivo, pois somente realizam a remoção da chave criptográfica. Entretanto, exigem grandes modificações na FTL, com a inclusão de algoritmos de criptografia e de geração e gerência de chaves. Também impõem uma penalidade nas operações de leitura e escrita em arquivos, pois os blocos devem ser decifrados antes de saírem do dispositivo e devem ser cifrados quando entram no dispositivo. Por estes motivos, o presente trabalho se concentrou nos métodos de remoção por sobrescrita e apagamento.

8.2 **Análise da resistência a ameaças**

Os métodos de remoção segura descritos neste trabalho são todos implementados na FTL, sendo portanto totalmente transparentes ao sistema operacional, ao sistema de arquivos e aos programas aplicativos. Além disso, não é objetivo destes métodos implementar um sistema de arquivos seguro, com confidencialidade, privacidade e controle de acesso. Seu único objetivo é remover arquivos de forma segura, ou seja, se um arquivo for removido não deve ser possível recuperar posteriormente o seu conteúdo, quer de forma total ou de forma parcial.

Desta forma, a única ameaça a ser analisada é a possibilidade de recuperação do conteúdo de um arquivo. Demais ameaças devem ser tratadas pelo sistema operacional, pois ocorrem externamente à memória *flash*.

Quando a remoção é realizada pelo sistema operacional, ou quando se utilizam aplicativos de sanitização e limpeza de disco, dados podem ser recuperados. Como demonstram os artigos e Wei e Swanson [59] [52], uma remoção através do sistema de arquivos permite recuperar até 91,3% dos dados em SSD e até 99,4% dos dados em *pendrive*. Mesmo com a utilização de programas específicos para remoção de arquivos, ainda é possível recuperar entre 5% a 67% dos dados. Estas técnicas, que utilizam rotinas do sistema operacional e do sistema de arquivos, falham por causa da complexidade do mapeamento realizado pela FTL, porque operações de sobrescrita em um mesmo bloco virtual acabam sendo distribuídas para vários blocos físicos distintos [59]. Observe-se que este mecanismo de mapeamento da FTL, projetado para uniformizar o desgaste da mídia, faz com que métodos de sobrescrita aplicáveis a discos magnéticos não funcionem com memória

flash. Por causa da FTL, as práticas forenses usuais de recuperação de dados em discos magnéticos não são efetivas para memória flash e precisam ser adaptadas [4].

Métodos de Sobrescrita e Apagamento, quando implementados pela FTL, garantem que nenhum bloco físico permaneça com o conteúdo original, porque ou é sobrescrito com zeros ou é apagado com uns. Mesmo que sejam utilizados circuitos especiais para realizar a leitura posterior contornando a FTL, como "Ming the Merciless" [59], até a data da realização deste trabalho, não é conhecido nenhum método que consiga recuperar dados de blocos sobrescritos ou apagados.

As normas do NIST recomendam duas passagens na realização de uma remoção segura, com padrões binários complementares, sem entretanto apresentar justificativa científica para esta recomendação. Como visto na Seção 4.2, o NIST inclusive recomenda uma única passagem para discos SSD, e duas passagens para dispositivos *flash* removíveis, sem novamente argumentar sobre esta diferenciação.

A maioria dos métodos analisados neste trabalho, inclusive o método ZO-ABP proposto, utiliza uma única passagem, com uma função de custo para decidir entre sobrescrita e apagamento. O único método que atende completamente as recomendações do NIST é o método de Huang. Entretanto, como visto no Capítulo 7, o método proposto por Huang é o que realiza a maior quantidade de operações sobre blocos e também a maior quantidade de apagamentos, o que acelera o desgaste do meio.

Caso a realização de duas passagens for desejável, pode-se adicionar sobrescrita de blocos antes de apagar uma unidade. Isto vale para todos os métodos analisados, exceto para o método de Huang, que já realiza as duas passagens. As consequências da sobrescrita adicional antes do apagamento podem ser vistos na Tabela 27. Esta tabela foi derivada da Tabela 14, onde cada apagamento realizado implica em 64 sobrescritas adicionais, porque cada unidade de apagamento contém 64 blocos.

Tabela 27 – Custo adicional de combinar sobrescrita com apagamento

Método	Apagamentos	Sobrescritas com zeros	Blocos operados	Sobrescritas adicionais	Total de Blocos	Aumento %
Normal	1271	0	81344	81344	162688	100,00
Zero-over	1271	220260	301604	81344	382948	26,97
Erase	6083	0	576290	389312	965602	67,55
ZO-A	1670	124927	231853	106880	338733	46,10
ZO-AB (Sun)	4510	21067	318857	288640	607497	90,52
ZOP-ABP	3889	21701	291371	248896	540267	85,42
ZO-ABP	3575	34380	272262	228800	501062	84,04
ZOP-AP	3083	49516	255888	197312	453200	77,11
ZO-AP	1670	124927	231853	106880	338733	46,10
ZOP-AB	4709	13339	335789	301376	637165	89,75
ZOP-A	3415	37602	265222	218560	483782	82,41
Huang	6083	220260	796550	0	796550	0,00

Blocos que são somente sobrescritos serão apagados quando da próxima reciclagem. Segundo estudos de [4], em discos SSD a reciclagem inicia em média após 200 s depois do disco ser energizado, e é completada após 350 s. Não foram encontrados dados sobre a ocorrência da reciclagem em dispositivos *flash* removíveis.

Métodos de Apagamento Criptográfico removem a chave, mas os blocos cifrados permanecem inalterados e podem ser lidos posteriormente. Como é chave é removida por sobrescrita (e adicionalmente por apagamento, no método de J. Lee), a probabilidade de recuperação desta chave é a mesma probabilidade de recuperação dos blocos nos Métodos de Sobrescrita e Apagamento. Entretanto, como os dados cifrados podem ser lidos, as seguintes ameaças adicionais devem ser consideradas:

- O algoritmo de criptografia utilizado deve resistir a ataques de força bruta;
- As chaves geradas devem ser fortes, no sentido que não serem facilmente deduzíveis;
- As chaves utilizadas devem ser randômicas, no sentido que uma análise das chaves não removidas não deve possibilitar deduzir as chaves já removidas;
- Os dados cifrados podem ser armazenados para uma possível decifragem posterior, se forem descobertas fraquezas futuras, quer no algoritmo de criptografia, quer no algoritmo de geração de randômicos, quer no método de gerência de chaves.

Considerando estas ameaças, Swanson e Wei propõe o método SAFE [52], que além de realizar a remoção segura das chaves, também efetua a remoção segura dos blocos cifrados.

Considerando estes fatores, e mais o fato que toda a implementação deve ser feita na FTL, métodos de Apagamento Criptográfico não apresentam atualmente um fator custo-benefício que ofereça vantagens significativas contra os Métodos de Sobrescrita e Apagamento.

8.3 **Análise do Método Proposto**

O método proposto é um método de Sobrescrita e Apagamento e, assim como o método de Sun, é um método híbrido, ou seja, para cada unidade de apagamento a ser tratada a decisão entre sobrescrever com zeros ou apagar a unidade é feita através de uma função de custo. Na seção 7.3 foi realizada uma análise sobre possíveis variações para esta função custo. Como resultado definiu-se para o método proposto os seguintes parâmetros:

- Para a operação de sobrescrita, o seu custo é igual ao tempo necessário para sobrescrever os blocos a serem removidos na unidade *i*:

  $Custo_{ZO}(i) = T_{zo} = N_{blocos\_a\_remover}(i) * E_t$

- Para a operação de sobrescrita, o seu custo inicial é igual ao tempo necessário para copiar os blocos válidos para outra unidade e a seguir realizar o apagamento da unidade *i*:

  $T_A = N_{blocos\_válidos} * (L_t + E_t) + A_t$

- Do custo inicial do apagamento subtrai-se um fator de Benefício, para representar que o apagamento produziu blocos livres, com exceção dos blocos válidos copiados para outra unidade (ou seja, o apagamento poupou uma futura reciclagem):

  $Benefício_A(i) = ((N_{blocos\_unidade} - N_{blocos\_válidos}(i))/ N_{blocos\_unidade}) * A_t$

- Entretanto, o apagamento é redundante, se existirem blocos livres na unidade, então ao custo inicial deve ser somado um fator de penalidade:

  $Penalidade_A(i) = N_{blocos\_livres}(i) * E_t$

- Assim, o custo de um apagamento é dado por:

  $Custo_A(i) = T_A - Benefício_A(i) + Penalidade_A(i)$

  ou seja

  $Custo_A(i) = N_{blocos\_válidos}(i) * ((L_t + E_t) + A_t/N_{blocos\_unidade}) + N_{blocos\_livres} * E_t$

Para cada unidade de apagamento que contém blocos do arquivo a remover, calcula-se $Custo_{ZO}(i)$ e $Custo_A(i)$, e a seguir realiza-se a operação de menor custo associado. Este método foi denominado de ZO-ABP e, conforme analisado na Seção7.3, foi o que produziu melhores resultados, ou seja, um equilíbrio de quantidade de sobrescritas e apagamentos realizados, assim como uma pequena quantidade de blocos livres apagados.

Como as funções utilizadas dependem de características tecnológicas de uma memória *flash*, ou seja, do tempo de leitura L_t, do tempo de escrita E_t, do tempo de apagamento A_t e do número de blocos por unidade $N_{blocos\_unidade}$, realizou-se nas seções 7.4 e 7.5 análises para verificar a influência destas características. Para os valores atualmente encontrados em memórias *flash*, o método ZO-ABP demonstrou ser bem estável, ou seja, seu desempenho não foi significativamente afetado por variações destas características.

Caso futuramente ocorram grandes mudanças nas características temporais das memórias *flash*, a função custo utilizada deverá ser reavaliada. Para tanto, basta realizar nova análise com o Simulador desenvolvido. O Simulador foi desenvolvido de forma modular, tornado relativamente simples definir e implementar novas funções de custo.

9 CONCLUSÕES E TRABALHOS FUTUROS

Com o crescente uso de meios digitais, o armazenamento de dados e o acesso controlado a estes dados são fatores importantes na gerência de arquivos. A remoção de dados armazenados em memória persistente é parte importante desta gerência, e, ao longo da história da computação, vários métodos foram desenvolvidos para assegurar a remoção segura de dados sensíveis, secretos ou confidenciais de empresas e agências governamentais. Com o crescimento explosivo de dispositivos móveis e o surgimento da Internet das Coisas (IoT), a remoção segura para proteger a privacidade de um usuário comum é uma característica altamente desejável.

Para dispositivos magnéticos, os métodos comumente empregados envolviam operações de sobrescrita dos blocos ocupados com padrões especificamente projetados para obscurecer dados remanescentes. Entretanto, meios de armazenamento baseados em memória não-voltátil, como a memória *flash*, diferem de meios magnéticos tanto na tecnologia usada para armazenar dados como nos algoritmos empregados para gerenciar e acessar estes dados. Através da Camada de Tradução da *Flash* (FTL), memórias *flash* incorporam um nível de indireção entre os blocos lógicos endereçados pelo sistema operacional e os blocos físicos que efetivamente contêm os dados. As diferenças entre meio magnético e memória *flash* fazem com que as técnicas tradicionais de remoção segura não sejam efetivas.

Outra característica de memórias *flash* é a necessidade tecnológica de apagar eletronicamente um bloco antes que ele possa ser utilizado (escrito) novamente. Assim, qualquer método que utilize a propriedade “escreva-sobre-dados-antigos” de meios magnéticos precisa ser modificado antes de ser aplicável em memórias *flash*.

A FTL gerencia o mapeamento de blocos lógicos para físicos, controla o desgate uniforme do meio e realiza a reciclagem (apagamento) de blocos obsoletos. A maneira atual como estas operações são implementadas na FTL impede que o sistema operacional acesse diretamente os blocos físicos, e por consequência qualquer método que deseja fazer a remoção segura de arquivos deve ser implementado na FTL.

Este trabalho analisou diversos métodos propostos na literatura para realizar remoção segura de arquivos. Duas metodologias principais foram analisadas em detalhe: remoção por Sobrescrita e Apagamento e remoção por Apagamento Criptográfico. O uso de Apagamento Criptográfico requer que sejam implementados na FTL tanto o algoritmo de criptografia como a gerência de chaves criptográficas, e devido à complexidade desta realização, o trabalho concentrou-se em métodos de Sobrescrita e Apagamento.

Para permitir a análise detalhada dos métodos de Sobrescrita e Apagamento, foi desenvolvido um simulador, que suporta não só os métodos referenciados na literatura, mas também diversas variações sobre estes métodos. Com a utilização do simulador, o método proposto durante a análise teórica dos métodos foi testado e refinado para atingir bom desempenho. Pelas análises realizadas, um método foi considerado como tendo bom desempenho se apresenta um equilíbrio entre Sobrescritas e Apagamentos, realiza uma quantidade relativamente pequena de operações sobre blocos e minimiza a quantidade de blocos livres que são prematuramente apagados.

O método inicialmente proposto, identificado pela sigla ZOP-ABP, foi refinado em função do desempenho observado no simulador, resultando no método ZO-ABP. Para decidir entre a realização de uma Sobrescrita ou um Apagamento, o método calcula o custo temporal da sobrescrita e compara com o custo temporal do apagamento, porém considerando que um apagamento tem o benefício de já realizar a reciclagem de blocos mas recebe uma penalidade se for aplicado a uma unidade de apagamento que contenha muitos blocos livres.

Pelas características tecnológicas atuais das memórias *flash*, o método ZO-ABP foi o que apresentou o melhor desempenho. Considerando o aspecto de apagar prematuramente blocos livres, o método ZO-ABP foi nitidamente superior ao método de Sun.

Caso ocorram mudanças tecnológicas nos tempos das operações de uma memória *flash*, uma nova análise de desempenho dos métodos pode ser facilmente realizada com o simulador desenvolvido. Basta simular o conjunto dos experimentos com os novos valores dos tempos.

O simulador também pode ser utilizado para realizar uma análise teórica mais abrangente sobre a influência dos tempos das operações da *flash*. Por exemplo, poderia ser simulada uma memória *flash* onde o tempo de um apagamento de uma unidade seja igual ao tempo de escrita de um bloco. Outras variações teóricas sobre os valores relativos das operações de leitura, escrita e apagamento também podem ser analisadas. A quantidade de blocos em uma unidade de apagamento também pode ser variada e analisada.

Atualmente a FTL recebe do sistema operacional operações a serem realizadas sobre blocos, e os métodos de remoção encontrados na literatura, assim como o método proposto, realizam a remoção somente com o conhecimento destes blocos. Esta remoção é realizada sem que a FTL tenha conhecimento da real dimensão do arquivo, ou seja, não consegue tomar decisões no nível de arquivo, mas somente no nível de blocos. Esta característica pode levar a decisões equivocadas, que podem comprometer o desempenho dos métodos de remoção. Por exemplo, quando se vai apagar uma unidade, os blocos ainda válidos desta unidade devem ser copiados para outra, onde exista espaço livre. Entretanto, esta outra unidade pode ser a próxima unidade a ser apagada, pois também contém vários blocos do arquivo a ser removido. Como a FTL não tem a visão completa do arquivo, ela não tem condições de detectar que está movendo blocos válidos para uma unidade que será apagada imediatamente a seguir. Para detectar tais situações, seria necessário investigar como a FTL poderia receber mais informações sobre o arquivo a ser removido, sem no entanto tornar a FTL dependente de um sistema de arquivos.

As técnicas de remoção de arquivos em meios magnéticos devem ser adaptadas para dispositivos com memória *flash*. De acordo com as normas de remoção do NIST, a popularização de dispositivos com memória *flash* leva a mudanças revolucionárias – em oposição a mudanças evolucionárias – quando se faz a migração de técnicas estabelecidas e consolidadas de armazenamento magnético para armazenamento em *flash*.

REFERÊNCIAS

[1] Advanced Micro Devices, “AMD DL160 and DL320 Series Flash: New Densities, New Features”. 2003. Capturado em: http://www.spansion.com/Support/Application Notes/AMD DL160 and DL320 Series Flash New Densities, New Features.pdf. Dezembro 2015.

[2] Ban, A. “Flash file system”. US Patent, no. 5404485, 1995.

[3] Bauer, S.; Priyantha, N. B. “Secure Data Deletion for Linux File Systems”. In: Proceedings of the Usenix Security Symposium, 2001, pp. 153–164.

[4] Bell, G. B.; Boddington, R. “Solid State Drives: The Beginning of the End for Current Practice in Digital Forensic Recovery?”, *The Journal of Digital Forensics, Security and Law*, vol. 5 (3), 2010, pp. 5.

[5] Boneh, D.; Lipton, R. “A revocable backup system”. In: Proceedings of the 6th USENIX UNIX Security Symposium, 1996, pp. 91-96.

[6] Breeuwsma, M.; Jongh, M.; Klaver, C.; van der Knijff, R.; Roeloffs, M. “Forensic data recovery from flash memory”. *Small Scale Digital Device Forensics Journal*, vol. 1, 2007, pp. 1-17.

[7] CyberScrub. Capturado em: http://www.cyberscrub.com, Dezembro 2015.

[8] DOE. “Media Sanitization Manual”. US Department of Energy, DOE M 205.1-6. Capturado em: http://www.directives.doe.gov. Junho 2016.

[9] Gal, E.; Toledo, S. “Algorithms and Data Structures for Flash Memories”. *ACM Computing Surveys,* vol. 37, 2005, pp. 138–163.

[10] Garfinkel, S. “Anti-Forensics: Techniques, Detection and Countermeasures”. In: Proceedings of 2nd International Conference on i-Warfare and Security, 2007, pp. 77-84.

[11] Garfinkel, S.; Shelat, A. “Remembrance of Data Passed: A Study of Disk Sanitization Practices”. *IEEE Security & Privacy*, January 2003, pp. 17–27.

[12] Gleixner, T.; Haverkamp, F.; Bityutskiy, A. “UBI - Unsorted Block Images”. 2006. Capturado em: http://linux-mtd.infradead.org/doc/ubidesign/ubidesign.pdf. Dezembro 2015.

[13] Gutmann, P. “Secure deletion of data from magnetic and solid-state memory”. In: Proceedings of the 6th USENIX UNIX Security Symposium, 1996, pp. 77-90.

[14] Gutmann, P. “Data Remanence in Semiconductor Devices” In: Proceedings of the 10th Conference on USENIX Security Symposium, 2001, pp. 4.

[15] Hao, F.; Clarke, D.; Zorzo, A. F. “Deleting Secret Data with Public Verifiability.” *IEEE Transactions on Dependable and Secure Computing*, vol. 13 (6), 2016, pp. 617–29.

[16] Harris, R. “Arriving at anti-forence consensus: Examining how to define and control the anti-forence problem”. Capturado em: http://www.dfrws.otg/2006/proceedings/6Harris.pdf. Dezembro 2015.

[17] Huang, N.; He, J.; Zhao, B. “Secure Data Sanitization for Android Device Users”. *International Journal of Security and its Applications*, vol. 9(5), 2015, pp. 61-68.

[18] Hughes, G.; Coughlin, T.; Commins, D. “Disposal of Disk and Tape Data by Secure Sanitization”. *IEEE Security & Privacy*, vol. 7(4), 2009.

[19] Intel Corporation. “Understanding the Flash Translation Layer (FTL) Specification”. Capturado em: http://www.eetasia.com/ARTICLES/2002MAY/2002MAY07_MEM_AN.PDF?SOURCES=DOWNLOAD. Abril 2016.

[20] Intel Corporation. “*Advantages of TRIM”. Capturado em:* http://www.intel.com/content/www/us/en/support/solid-state-drives/000006462.html. Junho 2016.

[21] IRIG 106. “The Standard for Digital Flight Data Recording, Chapter 10”. Capturado em: http://www.irig106.org/docs/106-07/chapter10.pdf. Junho 2016.

[22] Jensen, P. “System and method of erasing non-volatile recording media”. US Patent no. 2006/012023 A1, Jun. 8, 2006.

[23] Joel, R.; Srdjan, C., Basin, D. “Data Node Encrypted File System: Efficient Secure Deletion for Flash Memory.” In: Proceedings of the 21st USENIX Conference on Security Symposium, USENIX Association, 2012, pp. 17–17.

[24] Joulov N.; Zadok E. “Adding secure deletion to your favorite file system”. In: Proceedings of the 3rd International IEEE Security in Storage Workshop, 2005, pp. 63-70.

[25] Kalos, M. J.; Kubo, R. A.; Yanes, J. A. “System and method for implementing hard disk drive data clear and purge”. US Patent no. 2008/0028141 A1, Jan. 31, 2008.

[26] KillDisk. Capturado em: http://www.killdisk.com. Dezembro 2015.

[27] Kingston. “Flash Memory Guide”. Capturado em: http://www.kingston.com/flash_memory_guide. Novembro 2016.

[28] Lee, B.; Son, K.; Won, D.; Kim, S. “Secure Data Deletion for USB Flash Memory”. *Journal of Information Science and Engineering*, vol. 27, 2011, pp. 933-952.

[29] Lee, J.; Heo, J.; Cho, Y.; Hong, J.; Shin, S. “Secure deletion for NAND flash file system”. In Proceedings of ACM Symposium on Applied Computing, 2008, pp. 1710-1714.

[30] Lee, J.; Yi, S.; Heo, J.; Park, H.; Shin, S.Y.; Choo, Y. “An Efficient Secure Deletion Scheme for Flash File Systems”. *Journal of Information Science and Engineering*, vol. 26, 2010, pp. 27-38.

[31] Mallery, J. R. “Secure file deletion: Fact or fiction?” GSEC Practical Assignment Version 1.2e, 2006.

[32] Mary, B.; Asami, S.; Deprit, E.; Ousterhout, J.; Seltzer, M. . “Non-volatile memory for fast, reliable file systems”. In: Proceedings of the fifth international conference on Architectural support for programming languages and operating systems (ASPLOS V), Richard L. Wexelblat (Ed.). ACM, New York, NY, USA, 1992, pp. 10-22.

[33] McGlaun, S. “Report: 637,000 Notebooks Lost in U.S. Airports Every Year”. Capturado em: http://www.dailytech.com/Report+637000+Notebooks+Lost+in+US+Airports+Every+Year/article12288.htm. Dezembro 2015.

[34] Micron Technology, Inc. “NAND Flash 101 Technical Note”. Capturado em: https://www.micron.com/~/media/documents/products/technical-note/nandflash/tn2919_nand_101.pdf. Agosto 2016.

[35] Micron Technology, Inc. “NAND Flash Memory Features”. Capturado em: http://www.micron.com/products/nand-flash. Novembro 2016.

[36] Micron Technology, Inc. “Technical Note: Design and Use Considerations for NAND Flash Memory”. 2006.

[37] Microsoft. “BitLocker Drive Encryption Overview”. Capturado em: http://windows.microsoft.com/en-us/windows-vista/bitlocker-drive-encryption-overview. Dezembro 2015.

[38] Mittal, S.; Vetter, J.S. "A Survey of Software Techniques for Using Non-Volatile Memories for Storage and Main Memory Systems”, IEEE Transactions on Parallel and Distributed Systems, vol. 27.5, 2015, pp. 1537-1550.

[39] NISPON. “National Industrial Security Program Operating Manual Supplement”. Capturado em: http://www.fas.org/sgp/library/nispom_sup.pdf. Maio 2016.

[40] NIST. “Guidelines for Media Sanitization”. NIST Special Publication 800-88. Capturado em: http://dx.doi.org/10.6028/NIST.SP.800-88r1. Maio 2016.

[41] NSA/CSS. “Storage Device Sanitization Manual”. Capturado em: https://www.nsa.gov//resources/everyone/media-destruction/assets/files/storage-device-declassification-manual.pdf. Maio 2016.

[42] Oikawa, S.; Miki, S. "Future Non-volatile Memory Storage Architecture and File System Interface”. In: First International Symposium on Computing and Networking (CANDAR), 2013, Dec. 2013, pp. 389-392.

[43] PC Card Standard, “Volume 10: Media Storage Formats Specification”, 2001.

[44] Peterson. Z.N.J.; Burns. R.; Herring. J.; Stubblefield. A.; Rubin. A.D. “Secure Deletion for a Versioning File System”. In: Proceedings of the 4th Conference on USENIX Conference on File and Storage Technologies (FAST), 2005, vol. 4, pp. 143-154.

[45] Petitcolas; F. A. P.; Ross, J. A.; Kuhn, M. G. “Information Hiding-a Survey.” *Proceedings of the IEEE*, vol. 87 (7), 1999, pp. 1062–1078.

[46] Plumb, C. “shred(1) - Linux man page”. Página do comando shred. Capturado em: https://linux.die.net/man/1/shred. Dezembro 2015.

[47] Rogers, M. “Panel session at CERIAS 2006 Information Security Symposium”. Capturado em: http://www.cerias.purdue.edu/symposium/2006/materials/pdfs/antiforensics.pdf. Dezembro 2015.

[48] Silberschatz, A.; Galvin, P. B.; Gagne, G. 2012. *Operating System Concepts*. 9 edition. Hoboken, NJ: Wiley.

[49] Standard A. N. “Atis telecom glossary”, May 2013.

[50] Stanton; P.; Yurcik, W.; Brumbaugh, L. “Protecting Multimedia Data in Storage: A Survey of Techniques Emphasizing Encryption.” In: Electronic Imaging 2005, pp. 18–29.

[51] Sun, K.; Choi, J.; Lee, D.; Noh, S. “Models and Design of an Adaptive Hybrid Scheme for Secure Deletion of Data in Consumer Electronics”. *IEEE Transactions on Consumer Electronics*, vol. 54, 2008, pp. 100–104.

[52] Swanson; S.; Wei, M. “Safe: Fast, Verifiable Sanitization for SSDs“, Technical Report cs2011-0963, UCSD, 2010.

[53] Tal, A. "NAND vs. NOR flash technology: The designer should weigh the options when using flash memory”. Capturado em: http://www.electronicproducts.com/Digital_ICs/NAND_vs_NOR_flash_technology.aspx. Novembro 2015.

[54] Toshiba. “Flash Memory Semiconductor & Storage Solutions”. Capturado em: http://toshiba.semicon-storage.com/us/product/memory/nand-flash.html. Novembro 2016.

[55] TrueCrypt. Capturado em: http:// www.truecrypt.org. Dezembro 2015.

[56] US-CERT. “Computer Forensics”. Capturado em: http://www.us-cert.gov/reading_room/forensics.pdf. Novembro 2015.

[57] Vecchia, E. D.; Weber, D.; Zorzo, A. “Antiforense Digital: conceitos, técnicas, ferramentas e estudos de caso”. In: Minicursos do XIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais - SBSeg , 2013.

[58] Wang, Y.; Qin, Z.; Shao, Z.; Wang, Q.; Li, S.; Yang, L. T. “A Real-Time Flash Translation Layer for NAND Flash Memory Storage Systems”. *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 1, January-March 2016, pp. 17-28.

[59] Wei, M.; Grupp, L. M.; Spada, F. M.; Swanson, S. “Reliably Erasing Data from Flash-Based Solid State Drives”. In: Proceedings of the 9th USENIX conference on File and Storage Technologies, Berkeley, CA, USA, 2011, pp. 105–117.

[60] Wook, J. O. “Reverse Engineering Flash Memory for Fun and Benefit”. Capturado em: https://www.blackhat.com/docs/us-14/materials/us-14-Oh-Reverse-Engineering-Flash-Memory-For-Fun-And-Benefit-WP.pdf. Junho 2016.

[61] Wright, C. P.; Jay D.; Erez Z. “Cryptographic File Systems Performance: What You Don’t Know Can Hurt You.” In: Proceedings of the Second IEEE International Security in Storage Workshop, SISW’03, 2003, pp. 47–47.

APÊNDICE A – RESULTADOS DA SIMULAÇÃO

Cada experimento foi simulado em 12 cenários distintos:

1. Criação seguida da remoção de um mesmo arquivo, repetidas vezes.
2. Remoção de arquivo de tamanhos menores que uma unidade de apagamento.
3. Remoção de arquivo de tamanhos maiores que uma unidade de apagamento.
4. Remoção de arquivos pequenos (dezenas de KBytes).
5. Remoção de arquivos grandes (da ordem de MBytes).
6. Criação de arquivos até ocupar toda a memória *flash*, seguido da remoção aleatória de alguns arquivos.
7. Criação de arquivos até ocupar toda a memória *flash*, seguido da remoção de todos os arquivos.
8. Criação de arquivos até ocupar toda a memória *flash*, remoção de todos os arquivos, criação de novos arquivos até ocupar toda a memória novamente, seguida da remoção de todos os arquivos.
9. Criação de arquivos, seguido da remoção de todos os arquivos, na mesma ordem em que forma criados.
10. Criação de arquivos, seguido da remoção de todos os arquivos, em ordem aleatória.
11. Criação de arquivos, seguido da remoção de todos os arquivos, em nova ordem aleatória.
12. Criação de arquivos, seguido da remoção de todos os arquivos, em ordem inversa à da criação.

A execução conjunta de todos os cenários realiza a remoção de um total de 2.437 arquivos. Para a análise quantitativa realizada no Capítulo 7, o conjunto de todos os 12 cenários foi simulado 15 vezes, variando-se os tempos das operações e o tamanho da unidade de apagamento, conforme especificado nas tabelas a seguir.

Tabela A-1: Leitura=10, Escrita=50, Apagamento=3000, Unidade de 32 blocos

Total Arquivos	Método	Leituras	Escritas	Apagamentos	Sobrescritas com zeros	Marcados obsoletos	Livres apagados	Blocos operados
2437	Normal	0	0	2553	0	220260	0	81696
2437	Zero-over	0	0	2553	220260	220260	0	301956
2437	Erase	43591	43591	9284	0	0	33237	384270
2437	ZO-A	0	0	2553	220260	220260	0	301956
2437	ZO-AB (Sun)	1182	1182	7713	13084	13084	25561	262264
2437	ZOP-ABP	5270	5270	7386	9814	9814	10961	256706
2437	ZO-ABP	1182	1182	7025	17916	17916	3623	245080
2437	ZOP-AP	1180	1180	6076	34463	34463	847	231255
2437	ZO-AP	0	0	2553	220260	220260	0	301956
2437	ZOP-AB	5286	5286	7994	6631	6631	30347	273011
2437	ZOP-A	1180	1180	6312	29258	29258	3261	233602
2437	Huang	43591	43591	9284	220260	0	33237	604530

Tabela A-2: Leitura=10, Escrita=50, Apagamento=3000, Unidade de 64 blocos

Total Arquivos	Método	Leituras	Escritas	Apagamentos	Sobrescritas com zeros	Marcados obsoletos	Livres apagados	Blocos operados
2437	Normal	0	0	1271	0	220260	0	81344
2437	Zero-over	0	0	1271	220260	220260	0	301604
2437	Erase	93489	93489	6083	0	0	75563	576290
2437	ZO-A	23	23	1670	124927	124927	33	231853
2437	ZO-AB (Sun)	4575	4575	4510	21067	21067	64111	318857
2437	ZOP-ABP	10387	10387	3889	21701	21701	18712	291371
2437	ZO-ABP	4541	4541	3575	34380	34380	4449	272262
2437	ZOP-AP	4530	4530	3083	49516	49516	2651	255888
2437	ZO-AP	23	23	1670	124927	124927	33	231853
2437	ZOP-AB	10537	10537	4709	13339	13339	70813	335789
2437	ZOP-A	4530	4530	3415	37602	37602	10866	265222
2437	Huang	93489	93489	6083	220260	0	75563	796550

Tabela A-3: Leitura=10, Escrita=50, Apagamento=3000, Unidade de 128 blocos

Total Arquivos	Método	Leituras	Escritas	Apagamentos	Sobrescritas com zeros	Marcados obsoletos	Livres apagados	Blocos operados
2437	Normal	0	0	633	0	220260	0	81024
2437	Zero-over	0	0	633	220260	220260	0	301284
2437	Erase	253672	253672	4639	0	0	119860	1101136
2437	ZO-A	2229	2229	1419	96312	96312	25434	282402
2437	ZO-AB (Sun)	12025	12025	2806	37137	37137	127419	420355
2437	ZOP-ABP	20425	20425	2282	39076	39076	52321	372022
2437	ZO-ABP	12114	12114	2032	55143	55143	28656	339467
2437	ZOP-AP	11281	11281	1719	74555	74555	5830	317149
2437	ZO-AP	2229	2229	1052	122254	122254	1257	261368
2437	ZOP-AB	20768	20768	2935	26817	26817	135106	444033
2437	ZOP-A	11310	11310	2044	52297	52297	45147	336549
2437	Huang	253672	253672	4639	220260	0	119860	1321396

Tabela A-4: Leitura=25, Escrita=220, Apagamento=500, Unidade de 32 blocos

Total Arquivos	Método	Leituras	Escritas	Apagamentos	Sobrescritas com zeros	Marcados obsoletos	Livres apagados	Blocos operados
2437	Normal	0	0	2553	0	220260	0	81696
2437	Zero-over	0	0	2553	220260	220260	0	301956
2437	Erase	43591	43591	9284	0	0	33237	384270
2437	ZO-A	4617	4617	7894	7408	7408	27910	269250
2437	ZO-AB (Sun)	4628	4628	7953	7315	7315	29697	271067
2437	ZOP-ABP	5198	5198	7159	13224	13224	3868	252708
2437	ZO-ABP	4583	4583	7138	13662	13662	3781	251244
2437	ZOP-AP	4566	4566	7136	13648	13648	3740	251132
2437	ZO-AP	4566	4566	7136	13648	13648	3740	251132
2437	ZOP-AB	5274	5274	7987	6656	6656	30135	272788
2437	ZOP-A	4624	4624	7895	7398	7398	27935	269286
2437	Huang	43591	43591	9284	220260	0	33237	604530

Tabela A-5: Leitura=25, Escrita=220, Apagamento=500, Unidade de 64 blocos

Total Arquivos	Método	Leituras	Escritas	Apagamentos	Sobrescritas com zeros	Marcados obsoletos	Livres apagados	Blocos operados
2437	Normal	0	0	1271	0	220260	0	81344
2437	Zero-over	0	0	1271	220260	220260	0	301604
2437	Erase	93489	93489	6083	0	0	75563	576290
2437	ZO-A	9798	9798	4656	14029	14029	68167	331609
2437	ZO-AB (Sun)	9867	9867	4693	13934	13934	70459	334020
2437	ZOP-ABP	10300	10300	3681	28606	28606	5493	284790
2437	ZO-ABP	9646	9646	3662	29273	29273	4998	282933
2437	ZOP-AP	9571	9571	3654	29536	29536	4693	282534
2437	ZO-AP	9560	9560	3652	29884	29884	4596	282732
2437	ZOP-AB	10527	10527	4708	13350	13350	70759	335716
2437	ZOP-A	9855	9855	4656	14002	14002	68110	331696
2437	Huang	93489	93489	6083	220260	0	75563	796550

Tabela A-6: Leitura=25, Escrita=220, Apagamento=500, Unidade de 128 blocos

Total Arquivos	Método	Leituras	Escritas	Apagamentos	Sobrescritas com zeros	Marcados obsoletos	Livres apagados	Blocos operados
2437	Normal	0	0	633	0	220260	0	81024
2437	Zero-over	0	0	633	220260	220260	0	301284
2437	Erase	253672	253672	4639	0	0	119860	1101136
2437	ZO-A	19990	19990	2915	27441	27441	133324	440541
2437	ZO-AB (Sun)	20768	20768	2935	26817	26817	135106	444033
2437	ZOP-ABP	20395	20395	2116	47683	47683	31234	359321
2437	ZO-ABP	20296	20296	2109	48343	48343	30453	358887
2437	ZOP-AP	20167	20167	2108	48292	48292	30402	358450
2437	ZO-AP	19400	19400	2098	48753	48753	29988	356097
2437	ZOP-AB	20799	20799	2936	26733	26733	135203	444139
2437	ZOP-A	20764	20764	2922	26840	26840	133446	442384
2437	Huang	253672	253672	4639	220260	0	119860	1321396

Tabela A-7: Leitura=25, Escrita=220, Apagamento=1500, Unidade de 32 blocos

Total Arquivos	Método	Leituras	Escritas	Apagamentos	Sobrescritas com zeros	Marcados obsoletos	Livres apagados	Blocos operados
2437	Normal	0	0	2553	0	220260	0	81696
2437	Zero-over	0	0	2553	220260	220260	0	301956
2437	Erase	43591	43591	9284	0	0	33237	384270
2437	ZO-A	2813	2813	7295	12173	12173	12170	251239
2437	ZO-AB (Sun)	3922	3922	7927	7977	7977	29593	269485
2437	ZOP-ABP	5190	5190	7193	12422	12422	4927	252978
2437	ZO-ABP	3916	3916	7114	14286	14286	3725	249766
2437	ZOP-AP	3867	3867	7072	14600	14600	3576	248638
2437	ZO-AP	2811	2811	6978	16248	16248	3275	245166
2437	ZOP-AB	5282	5282	7992	6645	6645	30287	272953
2437	ZOP-A	3878	3878	7654	8873	8873	21124	261557
2256	Huang	40615	40615	8588	203940	0	30261	559986

Tabela A-8: Leitura=25, Escrita=220, Apagamento=1500, Unidade de 64 blocos

Total Arquivos	Método	Leituras	Escritas	Apagamentos	Sobrescritas com zeros	Marcados obsoletos	Livres apagados	Blocos operados
2437	Normal	0	0	1271	0	220260	0	81344
2437	Zero-over	0	0	1271	220260	220260	0	301604
2437	Erase	93489	93489	6083	0	0	75563	576290
2437	ZO-A	8310	8310	4186	17794	17794	40966	302318
2437	ZO-AB (Sun)	9236	9236	4669	14621	14621	69561	331909
2437	ZOP-ABP	10349	10349	3794	25137	25137	12645	288651
2437	ZO-ABP	9008	9008	3651	29753	29753	4901	281433
2437	ZOP-AP	8934	8934	3611	30632	30632	4539	279604
2437	ZO-AP	8207	8207	3595	32114	32114	3513	278608
2437	ZOP-AB	10527	10527	4708	13350	13350	70759	335716
2437	ZOP-A	9129	9129	4208	16981	16981	41660	304551
2437	Huang	93489	93489	6083	220260	0	75563	796550

Tabela A-9: Leitura=25, Escrita=220, Apagamento=1500, Unidade de 128 blocos

Total Arquivos	Método	Leituras	Escritas	Apagamentos	Sobrescritas com zeros	Marcados obsoletos	Livres apagados	Blocos operados
2437	Normal	0	0	633	0	220260	0	81024
2437	Zero-over	0	0	633	220260	220260	0	301284
2437	Erase	253672	253672	4639	0	0	119860	1101136
2437	ZO-A	18081	18081	2637	30803	30803	100969	404501
2437	ZO-AB (Sun)	19854	19854	2929	27440	27440	135252	442060
2437	ZOP-ABP	20454	20454	2118	47509	47509	31363	359521
2437	ZO-ABP	19494	19494	2098	48774	48774	29959	356306
2437	ZOP-AP	19334	19334	2081	49283	49283	29322	354319
2437	ZO-AP	17521	17521	1994	55664	55664	19701	345938
2437	ZOP-AB	20799	20799	2936	26733	26733	135203	444139
2437	ZOP-A	19836	19836	2668	28852	28852	103152	410028
2437	Huang	253672	253672	4639	220260	0	119860	1321396

Tabela A-10: Leitura=25, Escrita=300, Apagamento=2000, Unidade de 32 blocos

Total Arquivos	Método	Leituras	Escritas	Apagamentos	Sobrescritas com zeros	Marcados obsoletos	Livres apagados	Blocos operados
2437	Normal	0	0	2553	0	220260	0	81696
2437	Zero-over	0	0	2553	220260	220260	0	301956
2437	Erase	43591	43591	9284	0	0	33237	384270
2437	ZO-A	3322	3322	7330	11419	11419	12699	252623
2437	ZO-AB (Sun)	3932	3932	7928	7964	7964	29615	269524
2437	ZOP-ABP	5190	5190	7193	12422	12422	4927	252978
2437	ZO-ABP	3916	3916	7114	14286	14286	3725	249766
2437	ZOP-AP	3867	3867	7072	14600	14600	3576	248638
2437	ZO-AP	3317	3317	6993	15749	15749	3281	246159
2437	ZOP-AB	5286	5286	7994	6631	6631	30347	273011
2437	ZOP-A	3884	3884	7656	8854	8854	21118	261614
2437	Huang	43591	43591	9284	220260	0	33237	604530

Tabela A-11: Leitura=25, Escrita=300, Apagamento=2000, Unidade de 64 blocos

Total Arquivos	Método	Leituras	Escritas	Apagamentos	Sobrescritas com zeros	Marcados obsoletos	Livres apagados	Blocos operados
2437	Normal	0	0	1271	0	220260	0	81344
2437	Zero-over	0	0	1271	220260	220260	0	301604
2437	Erase	93489	93489	6083	0	0	75563	576290
2437	ZO-A	8310	8310	4186	17794	17794	40966	302318
2437	ZO-AB (Sun)	9861	9861	4692	13941	13941	70408	333951
2437	ZOP-ABP	10349	10349	3794	25137	25137	12645	288651
2437	ZO-ABP	9646	9646	3662	29273	29273	4998	282933
2437	ZOP-AP	9564	9564	3626	30240	30240	4167	281432
2437	ZO-AP	8207	8207	3595	32114	32114	3513	278608
2437	ZOP-AB	10523	10523	4710	13312	13312	70891	335798
2437	ZOP-A	9730	9730	4231	16281	16281	42459	306525
2437	Huang	93489	93489	6083	220260	0	75563	796550

Tabela A-12: Leitura=25, Escrita=300, Apagamento=2000, Unidade de 128 blocos

Total Arquivos	Método	Leituras	Escritas	Apagamentos	Sobrescritas com zeros	Marcados obsoletos	Livres apagados	Blocos operados
2437	Normal	0	0	633	0	220260	0	81024
2437	Zero-over	0	0	633	220260	220260	0	301284
2437	Erase	253672	253672	4639	0	0	119860	1101136
2437	ZO-A	18909	18909	2652	29884	29884	102118	407158
2437	ZO-AB (Sun)	19994	19994	2928	27396	27396	134984	442168
2437	ZOP-ABP	20950	20950	2122	46808	46808	31581	360324
2437	ZO-ABP	19570	19570	2099	48774	48774	29960	356586
2437	ZOP-AP	19334	19334	2081	49283	49283	29322	354319
2437	ZO-AP	18352	18352	2000	55258	55258	19829	347962
2437	ZOP-AB	21774	21774	2951	25812	25812	136177	447088
2437	ZOP-A	19962	19962	2665	28868	28868	102642	409912
2437	Huang	253672	253672	4639	220260	0	119860	1321396

Tabela A-13: Leitura=30, Escrita=250, Apagamento=3000, Unidade de 32 blocos

Total Arquivos	Método	Leituras	Escritas	Apagamentos	Sobrescritas com zeros	Marcados obsoletos	Livres apagados	Blocos operados
2437	Normal	0	0	2553	0	220260	0	81696
2437	Zero-over	0	0	2553	220260	220260	0	301956
2437	Erase	43591	43591	9284	0	0	33237	384270
2437	ZO-A	1876	1876	6888	17486	17486	4807	241654
2437	ZO-AB (Sun)	3354	3354	7889	8735	8735	28971	267891
2437	ZOP-ABP	5224	5224	7274	11213	11213	7433	254429
2437	ZO-ABP	3336	3336	7107	14487	14487	4041	248583
2437	ZOP-AP	3306	3306	6982	15853	15853	3281	245889
2437	ZO-AP	1876	1876	6738	20845	20845	2570	240213
2437	ZOP-AB	5282	5282	7992	6645	6645	30287	272953
2437	ZOP-A	3316	3316	7162	12679	12679	8256	248495
2437	Huang	43591	43591	9284	220260	0	33237	604530

Tabela A-14: Leitura=30, Escrita=250, Apagamento=3000, Unidade de 64 blocos

Total Arquivos	Método	Leituras	Escritas	Apagamentos	Sobrescritas com zeros	Marcados obsoletos	Livres apagados	Blocos operados
2437	Normal	0	0	1271	0	220260	0	81344
2437	Zero-over	0	0	1271	220260	220260	0	301604
2437	Erase	93489	93489	6083	0	0	75563	576290
2437	ZO-A	6514	6514	3885	22581	22581	24744	284249
2437	ZO-AB (Sun)	8555	8555	4645	15421	15421	68706	329811
2437	ZOP-ABP	10364	10364	3805	24553	24553	13331	288801
2437	ZO-ABP	8320	8320	3640	30336	30336	4891	279936
2437	ZOP-AP	8205	8205	3585	31552	31552	3978	277402
2437	ZO-AP	6490	6490	3524	34452	34452	3330	272968
2437	ZOP-AB	10537	10537	4709	13339	13339	70813	335789
2437	ZOP-A	8292	8292	3988	19513	19513	28585	291329
2437	Huang	93489	93489	6083	220260	0	75563	796550

Tabela A-15: Leitura=30, Escrita=250, Apagamento=3000, Unidade de 128 blocos

Total Arquivos	Método	Leituras	Escritas	Apagamentos	Sobrescritas com zeros	Marcados obsoletos	Livres apagados	Blocos operados
2437	Normal	0	0	633	0	220260	0	81024
2437	Zero-over	0	0	633	220260	220260	0	301284
2437	Erase	253672	253672	4639	0	0	119860	1101136
2437	ZO-A	15862	15862	2550	33892	33892	92097	392016
2437	ZO-AB (Sun)	18169	18169	2902	29349	29349	133511	437143
2437	ZOP-ABP	20399	20399	2118	47575	47575	31424	359477
2437	ZO-ABP	18012	18012	2085	49899	49899	29531	352803
2437	ZOP-AP	17521	17521	2066	50408	50408	28911	349898
2437	ZO-AP	15682	15682	1953	58917	58917	17277	340265
2437	ZOP-AB	20799	20799	2936	26733	26733	135203	444139
2437	ZOP-A	18068	18068	2603	31109	31109	96630	400429
2437	Huang	253672	253672	4639	220260	0	119860	1321396