Eliminação Segura de Arquivos em Memória Não-Volátil

Julia S. Weber[1], Avelino F. Zorzo[1]

[1]Faculdade de Informática –PUCRS – Porto Alegre – RS - Brasil

julia.weber@acad.pucrs.br, avelino.zorzo@pucrs.br

Abstract. *The Internet of Things (IoT) and the popularization of mobile devices bring 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. Consequently, new methods of safe removal have been developed, which employ Zero Override and Block Erase. This work analyzes these methods and proposes a new one, with better performance. The proposed method is hybrid, combining overwriting and deletion operations to obtain a balanced use of these operations, avoid unnecessary deletion of unused blocks and reduce premature memory wear.*

Resumo. *A Internet das Coisas (IoT) e a popularização de dispositivos móveis traz novos desafios quanto a remoção de arquivos. Técnicas tradicionalmente empregadas em meios magnéticos não são efetivas para memórias não voláteis, como a memória flash. Consequentemente, novos métodos de remoção segura foram desenvolvidos, que empregam operações de Sobrescrita com Zeros e de Apagamento de Blocos. Este trabalho analisa estes métodos e propõe um novo, com melhor desempenho. O método proposto é um híbrido, combinando 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.*

1. Introdução

Memória não-volátil é um tipo especial de memória que pode reter informações mesmo em caso de perda de energia. 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 acessada mais rapidamente do que os discos rígidos tradicionais. Com o crescente uso de memória *flash* em celulares, *smartphones, tablets* e *pen drives*, cresce também a necessidade de garantir a proteção dos dados armazenados. Muitos arquivos nestes dispositivos contêm dados pessoais e privados, que não devem ser acessados por pessoas não autorizadas, o que pode acontecer em caso de roubo ou perda do dispositivo, inclusive para arquivos que foram removidos. Arquivos removidos não devem ser recuperados por terceiros, pois isto também compromete a privacidade.

Dispositivos de armazenamento são fundamentais em sistemas de computação, e uma regra básica dos projetistas 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 resguardar 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 [13].

Eliminação segura de dados consiste em suprimir permanentemente os dados digitais de um meio físico de tal modo que os dados se tornem irrecuperáveis [7]. Assim, por exemplo, métodos antiforenses [6] são utilizados para eliminar permanentemente arquivos específicos ou sistemas de arquivos inteiros. Para a eliminação de dados, algumas soluções usam deleção por sobrescrita (*overwriting*) [2] [7] [8], sendo estas as mais comuns no momento, ou mesmo por criptografia [9].

Contudo, isso dá origem a outro problema. O uso de dispositivos com memórias não voláteis aumentou significativamente nos últimos anos. 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 [16]. 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.

Assim, este artigo apresenta uma proposta para remoção segura de arquivos em memórias não-voláteis que emprega operações de sobrescrita com zeros e de apagamento de blocos. O método proposto é híbrido, combinando 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.

Este artigo está organizado da seguinte forma: a Seção 2 apresenta as características de memórias *flash*; a Seção 3 descreve alguns dos principais métodos de remoção segura de dados em memórias não-voláteis, e apresenta o método proposto; a Seção 4 apresenta os experimentos realizados para comparar métodos no estado da arte com o método proposto; e, a Seção 5 apresenta as conclusões deste trabalho.

2. Características de memórias *flash*

A memória *flash* é um meio de armazenamento não-volátil que consiste em uma série de componentes eletrônicos que armazenam informações [1]. 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 1 ilustra como a arquitetura de uma memória *flash* é relacionada com aplicativos e sistemas de arquivos.

Sistemas de arquivo são normalmente implementados em memória *flash* com tecnologia NAND. A memória *flash* NAND é dividida em dois níveis de granularidade. O primeiro é chamado nível de unidades de apagamento, que são da ordem de 64KB ou 128 KB [13] 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 [5]. 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.

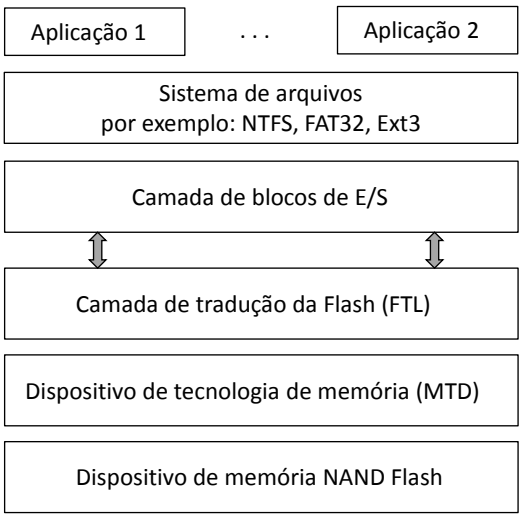


Figura 1 – Arquitetura de uma memória *flash* do tipo NAND

Apagar memória *flash* causa desgaste físico significativo [11]. Blocos de *flash* toleram entre 10^4 a 10^5 apagamentos antes que se tornem blocos não confiáveis [3]. 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* normalmente consiste de duas camadas, a FTL (Flash Translation Layer) e a MTD (Memory Technology Device), como ilustrado na Figura 1. 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 as funções primitivas na memória *flash*, funções de escrever, ler e apagar.

Memória *flash* apenas permite dois estados: apagado e não apagado. No estado de apagado, um byte de uma memória flash NAND tem todos os bits ligados (0xFF). Um bit em "1" (apagado) pode ser escrito e transformado em "0" (não apagado), mas não pode voltar para um. Para retornar o bit para o seu estado de apagado, um bloco significativamente maior da mídia *flash*, chamado de unidade de apagamento deve ser apagada. 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. A FTL também cuida de recuperar os blocos de dados descartados para reutilização.

Um bloco de memória *flash* é a unidade básica para a operação de leitura/escrita, enquanto uma unidade é a quantidade mínima para uma operação de apagamento. O desempenho destas três operações é bastante diferente, como mostrado na Tabela 1.

Uma operação de apagamento leva muito mais tempo do que uma escrita ou de leitura. Com a realização de operações de escrita na memória *flash*, o espaço livre diminui e operações de coleta de lixo são invocadas para recuperar algum espaço livre para reutilização.

Tabela 1 – Especificações de Memória flash NAND [15]

Características	SLC NAND Flash	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. Métodos para remoção segura

O ponto deste trabalho é a 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 foi objetivo implementar ou analisar sistemas de arquivos seguros, mas sim realizar a remoção segura de arquivos, limitou-se a pesquisa a métodos de eliminação por sobrescrita.

O problema causado por dados que permanecem no meio de armazenamento foi descoberto pela primeira vez em meio magnético [7]. Mesmo se a informação é sobrescrita 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.

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 aqui uma análise para este caso, pois o seu custo fornece um valor teórico ideal para métodos de sobrescrita.

Independente do sistema de arquivos utilizado pelo sistema operacional, 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.

Para análise de custos são utilizadas as seguintes grandezas:

- L_t : Tempo de uma leitura (valor de referência básica).
- E_t : Tempo de uma escrita (varia de 8 a 11 vezes o tempo de uma leitura).
- A_t : Tempo de um apagamento (varia de 16 a 100 vezes o tempo de uma leitura).

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, 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.

Técnicas de remoção segura que são geralmente utilizadas para discos magnéticos não são aplicáveis para a memória *flash*. Devido a essa limitação, existem alguns métodos existentes para a eliminação segura de memória *flash*, detalhados nas próximas seções.

3.1. Método de apagamento híbrido (Método de Sun)

Sun [14] propôs um método híbrido adaptativo que combina sobrescritas com zero (*zero-overwriting*) e apagamento de blocos (*erase blocks*). Sobrescritas com zero é um método de eliminação que substitui dados nos blocos por 0x00 para que os dados existentes possam ser apagados de forma segura. Em contraste, apagamento de blocos 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.

O esquema híbrido adaptativo proposto por Sun [14] 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.

Assim, o custo de uma sobrescrita em uma unidade *i* 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 custo de apagamento é o tempo necessário para copiar os blocos ainda válidos para outra unidade de apagamento e a seguir apagar a unidade *i*:

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

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

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

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

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

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

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

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

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:

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

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

E para o apagamento tem-se:

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

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

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

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 por Sun, entretanto, realiza a comparação dos dois para cada unidade de apagamento *i*, e usa nesta unidade a operação de menor custo.

3.2. Método de apagamento duplo (Método de Huang)

Huang [10] 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 [4], [12].

Se por um lado o método segue 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.

O método duplo de Huang [10] realiza tanto uma operação de sobrescrita como uma operação de apagamento, e portanto o seu tempo é dado pelo somatório das operações em todas unidades de apagamento que contêm blocos do arquivo a ser removido.

3.3. Métodos propostos (ZOP-EBP e ZO-EBP)

O método híbrido de remoção de arquivos proposto por Sun utiliza uma função de custo para decidir entre uma sobrescrita ou 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 ocorre com blocos sobrescritos.

Entretanto, o método de Sun não considera a existência de setores livres 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.

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. Por considerar que blocos livres sendo 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:

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

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, aqui 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

O método proposto é um método híbrido de Sobrescrita e Apagamento e utiliza uma função de custo para decidir entre sobrescrever com zeros ou apagar a unidade. 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 prejudicial 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 $Custo_A(i) = N_{blocos\_válidos}(i) * ((L_t + E_t) + A_t/N_{blocos\_unidade}) + N_{blocos\_livres} * E_t$

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. Estes métodos foram denominados de ZOP-EBP, que considera as duas penalidades, e ZO-EBP, que considera somente a penalidade de apagamento. A próxima seção apresenta experimentos descrevendo as vantagens e desvantagens de cada método proposto.

4. Experimentos

Para realizar uma análise quantitativa e qualitativa dos métodos estudados, bem como uma análise dos aperfeiçoamentos propostos foi implementado um simulador na linguagem de programação C. Este simulador foi desenvolvido para simular as operações básicas de um sistema de arquivos, particularmente a criação e remoção de arquivos. Foram simuladas as operações da FTL, ou seja, mapeamento de blocos virtuais para físicos, gerência do estado de blocos físicos e *garbage collector*, assim como as operações sobre blocos físicos: leitura, escrita, sobrescrita e apagamento. Além disto, foram simulados os diversos métodos de remoção segura de arquivos.

Para comparar os vários métodos de remoção, são contabilizadas as operações de apagamentos, sobrescritas com zeros, blocos livres apagados, leituras e escritas de blocos, assim como a quantidade total de blocos operados em remoção de arquivo.

Com o uso do Simulador foi possível validar os métodos de remoção 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. Para os experimentos realizados, foram empregados os mesmos tempos de 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 contabilizados os blocos copiados entre unidade de apagamento (leituras e escritas), as unidades apagadas (apagamentos), os blocos sobrescritos (sobrescritas com zeros), os blocos livres que foram prematuramente apagados (livres apagados) e o total de blocos operados, quer por leitura, escrita, sobrescrita ou apagamento (blocos operados).

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.

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.

Os métodos de sobrescrita com zeros (*zero-overwrite*) e de apagamento (*erase*) foram simulados somente para estabelecer os limites superiores das operações de sobrescrita e de apagamento, respectivamente. A partir destes limites pode-se estimar o desempenho de cada método, ou seja, as reduções nas quantidades de operações necessárias.

Tabela 2 – Resultados da simulação do refinamento do método proposto

Método	Apagamentos		Sobrescritas com zeros		Livres apagados		Blocos operados	
Zero-over	1271	20,89%	220260	100%	0	0%	301604	37,86%
Erase	6083	100%	0	0%	75563	100%	576290	72,35%
Sun	4510	74,14%	21067	9,56%	64111	84,84%	318857	40,03%
ZOP-EBP	3889	63,93%	21701	9,85%	18712	24,78%	291371	36,58%
ZO-EBP	3575	58,77%	34380	15,61%	4449	5,89%	272262	34,18%
Huang	6083	100%	220260	100%	75563	100%	796550	100%

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. Os apagamentos realizados reciclaram prematuramente 64.111 blocos livres. 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 ZOP-EBP 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. Mesmo assim, a quantidade total de operações realizadas foi de 291.371, 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). Por causa disto, é o método que envolve a maior quantidade de blocos operados, com um total de 796.550 blocos.

Apesar do desempenho do método proposto ZOP-EBP 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. 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. O método proposto originalmente é identificado pela sigla ZOP-EBP, onde a letra P indica a Penalidade tanto na sobrescrita com zeros (ZO) como no apagamento (E). O Segundo método é identificado pela sigla ZO-EBP, 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).

Como pode ser verificado na Tabela 2, uma análise comparativa do método de Sun, o primeiro método proposto (sigla ZOP-EBP) e o segundo método proposto (sigla ZO-EBP) mostra que o método ZO-EBP 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-EBP.

O método ZO-EBP 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

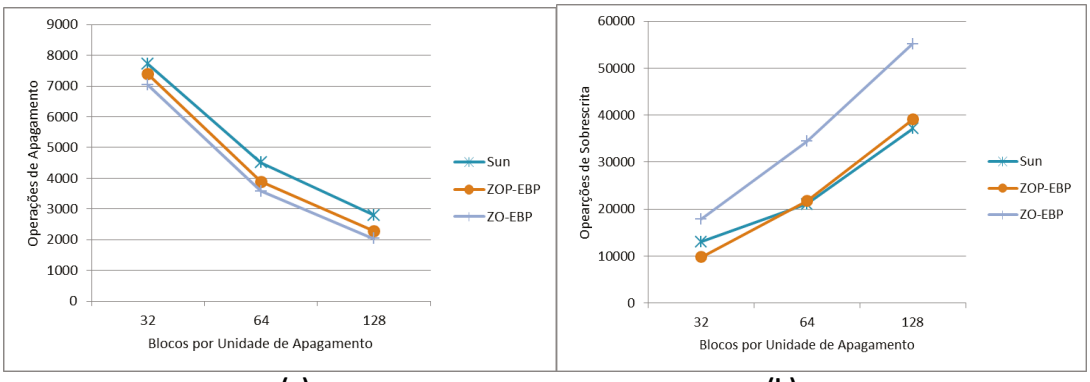
ZOP-EBP. O método ZO-EBP, entretanto, apresentou um aumento considerável das operações de sobrescrita, realizando 34.380 sobrescritas com zeros, contra 21.067 sobrescritas do método de Sun e 21.701 do método ZOP-EBP. Este aumento era esperado, pois a retirada do fator de penalidade da operação de sobrescrita 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.

Finalmente, o método ZO-EBP 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-EBP. Como o limite inferior de blocos operados é de 220.260, definido pelo método de sobrescrita, o método ZO-EBP foi dentre os três o que mais se aproximou deste limite.

Por estes resultados, o novo método proposto ZO-EBP foi o que apresentou os melhores resultados. Em termos percentuais, é o que apresentou a maior redução nas operações de apagamento (58,77%) e apagou o menor percentual de blocos livres (5,89%). Perde para o método de Sun e o método ZOP-EB em relação à redução de sobrescritas (15,61% contra 9,56% e 9,85%), mas esta desvantagem é compensada por ser o método que operou sobre o menor percentual de blocos (34,18%).

4.1 Análise da influência do tamanho da unidade de apagamento

Memórias *flash* atualmente têm unidades de apagamento que agrupam 32, 64 ou 128 blocos, 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.



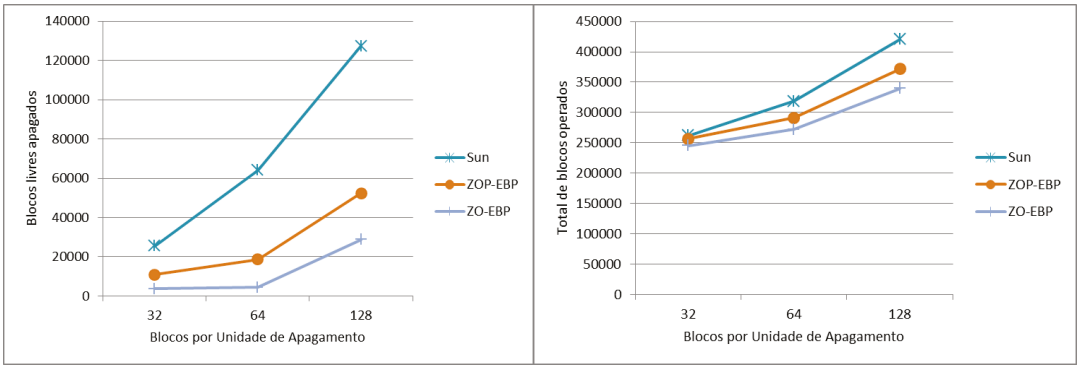
(a) **(b)**

Figura 2 – Quantidade de apagamentos e sobrescritas por tamanho de unidade

Como pode ser observado na Figura 2a, a quantidade de apagamentos realizada diminui conforme o tamanho da unidade aumenta. Este resultado é esperado, pois se a unidade de apagamento contém mais blocos, os arquivos de mesmo tamanho tendem a ocupar menos unidades de apagamento. Por outro lado, como visto na Figura 2b, a quantidade de blocos sobrescritos aumenta conforme o tamanho da unidade também aumenta. Este resultado comprova que os métodos garantem um bom balanceamento entre apagamentos e sobrescritas, pois a redução das operações de apagamento implica

em um aumento correspondente nas operações de sobrescrita. Todos os métodos apresentam o mesmo comportamento conforme o tamanho da unidade de apagamento aumenta.

A Figura 3a 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. 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 considera uma desvantagem apagar blocos livres, e em consequência os métodos "EBP" apresentam desempenho melhor neste aspecto. Embora os métodos apresentem o mesmo comportamento, pode-se verificar que o método de Sun apresenta uma quantidade significativamente maior de blocos livres apagados. A Figura 3b mostra a quantidade total de setores operados (apagados, sobrescritos ou copiados) em cada um dos métodos. Como pode ser visto, esta quantidade aumenta em função do tamanho da unidade de apagamento. Todos os métodos variam da mesma forma, e o método de Sun tem o maior aumento, pelo fato de apagar uma quantidade significativa de setores livres.



(a) **(b)**

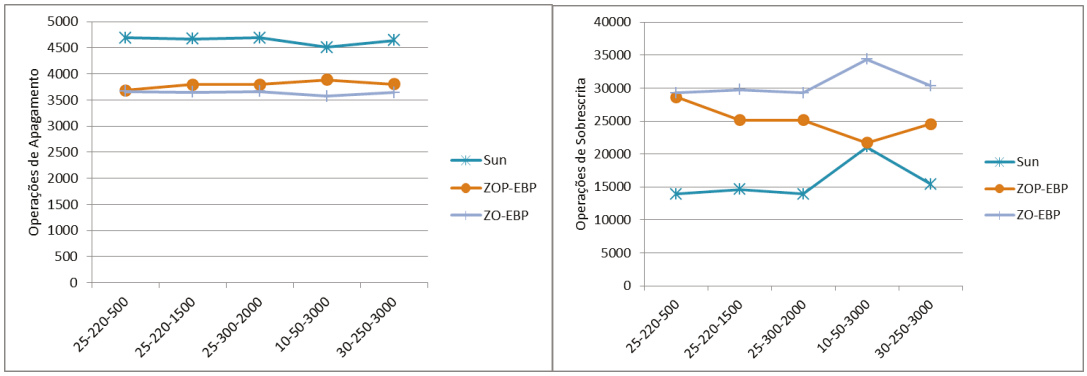
Figura 3 – Blocos livres apagados e blocos operados por tamanho de unidade

4.2 Análise da influência do tempo das operações

Os três métodos escolhidos 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*:

- 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 Figura 4a mostra a quantidade de operações de apagamentos realizadas por cada um dos três métodos. Não são notadas variações significativas em cada um dos métodos, indicando que os métodos permanecem relativamente constantes em função da variação dos tempos das operações.

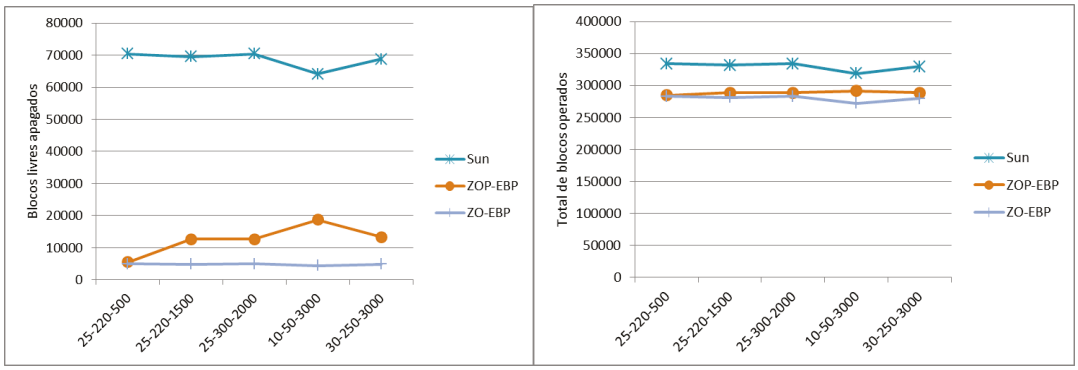


(a) (b)

Figura 4 – Quantidade de apagamentos e sobrescrita por tempos de operações

Na Figura 4b são mostradas as quantidades de operações de sobrescrita de blocos realizadas por cada um dos três métodos. Em contraste com o número de operações de apagamento, agora existem variações significativas em alguns métodos.

Em relação à quantidade de blocos livres apagados prematuramente pelo apagamento da unidade em que se encontram, a Figura 5a mostra como se comportam cada um dos métodos. O método de Sun não sofre muita variação, mas é o que mais desperdiça blocos livres, justamente por não considerar este tipo de bloco nas suas funções de custo. O método ZO-EBP é o mais independente dos tempos das operações, além de apresentar as menores quantidades de blocos livres que são apagados.



(a) (b)

Figura 5 – Blocos livres apagados e total de blocos operados por tempos de operações

A Figura 5b apresenta a quantidade total de blocos operados. Não se observam variações significativas em função dos tempos das operações.

5. Conclusão

Pelas análises realizadas, um método é 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-EBP, foi refinado em função do desempenho observado no simulador, resultando no método ZO-EBP. 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. Estes dois métodos apresentam desempenho melhor do que os métodos atualmente descritos na literatura especializada [10] e [14], além de, comparativamente, reduzirem o desgaste da mídia.

6. Referências

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

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

[3] 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.

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

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

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

[7] 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.

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

[9] Hao, F.; Clarke, D.; Zorzo, A.F. “Deleting secret data with public verifiability”, IEEE Transactions on Dependable and Secure Computing, vol. 13, num. 6, 2016, pp. 617-629.

[10] 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.

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

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

