Centro Universitário de Brasília

Artigo para a obtenção da Menção I

Robson Lucas Lopes Teixeira de Carvalho

DIFERENÇAS ENTRE O GERENCIAMENTO DE MEMÓRIA NO LINUX E WINDOWS

Brasília

2020

Robson Lucas Lopes Teixeira de Carvalho

DIFERENÇAS ENTRE O GERENCIAMENTO DE MEMÓRIA NO LINUX E WINDOWS

Artigo apresentado ao Centro Universitário de Brasília (UniCEUB) como requisito para a obtenção da menção I no curso Ciência da Computação, na disciplina Sistemas Operacionais.

Professor: Aderbal Botelho Leite Neto

Brasília

2020

DIFERENÇAS ENTRE O GERENCIAMENTO DE MEMÓRIA NO LINUX E WINDOWS

RESUMO

A Memória é um recurso importante que deve ser bem gerenciado. Enquanto a capacidade de armazenamento dos computadores vem crescendo continuamente, a complexidade do software cresce à taxas maiores. A parte do sistema operacional que gerencia a memória é chamada de gerenciador de memória, sendo o objeto deste artigo. Dentre outras tarefas, o gerenciador de memória monitora quais partes da memória estão em uso e quais estão disponíveis; aloca e libera memória para os processos; e gerencia a troca de processos entre memória principal e secundária (quando a memória principal não é capaz de abrigar todos os processos). Nesse cenário, os sistemas operacionais utilizados como referência foram o Linux e o Windows 8.

Palavras-chave: Sistema operacional, Windows, Linux.

1 Introdução

A memória principal (RAM) é um recurso fundamental para o funcionamento de um computador e deve ser bem gerenciado. Atualmente, mesmo que o computador pessoal consiga ter milhares de vezes mais memória do que os computadores das primeiras gerações, os programas passaram a ficar cada vez maiores e mais rápidos do que as memórias.

Dentro da Computação, existe o conceito da hierarquia de memória, em que o computador possui uma pequena quantidade de memória cache, muito rápida, cara e volátil, uma quantidade de memória principal (RAM) de velocidade e custo médios e uma quantidade muito grande de memória de armazenamento em disco (HD), considerada lenta e barata, sem mencionar o armazenamento removível como os dispositivos USB ou DVDs. Portanto, o problema principal para o gerenciamento de memória é que os programas atuais são muito grandes para rodarem, inteiramente, na memória cache.

A cada dia que passa, os profissionais de TI e usuários comuns precisam de mais memória para rodar mais programas simultaneamente para poderem

tratar de mais informações. Porém, um tratamento mais eficiente da memória utilizada não é de fácil implementação. Logo, o gerenciamento de memória é responsabilidade do sistema operacional, conforme Tanenbaum (2016, p. 125) afirma:

> A parte do sistema operacional que gerencia (parte da) hierarquia de memórias é chamada de gerenciador de memória. Sua função é gerenciar eficientemente a memória: controlar quais partes estão sendo usadas, alocar memória para processos quando eles precisam dela e liberá-la quando tiverem terminado.

2 Gerenciamento de memória no Linux

O Linux é um tipo de sistema operacional caracterizado por ter memória virtual paginada, ou seja, o programa em execução cujo tamanho seja maior que a memória física pode estar apto para ser executado. Portanto, o sistema operacional é o grande responsável por alimentar a memória com as partes dos programas que estão efetivamente em uso, deixando o resto no disco rígido. Dessa forma, “O modelo de memória do Linux é direto, a fim de permitir a portabilidade dos programas e tornar possível implementar o Linux em máquinas com unidades de gerenciamento de memória amplamente diferentes. ” (TANENBAUM, 2016, p. 520).

A título de exemplo, um programa com 16MB de memória pode rodar em um computador de 4MB de memória, sendo parte do programa dinamicamente carregado em memória de acordo com a necessidade de execução, com as suas partes sendo copiadas do disco rígido para a memória e vice-versa, se for necessário. Entretanto, o uso da memória virtual fará com que o computador fique mais lento, mesmo que ele aparente incluir mais memória RAM do que verdadeiramente possui.

No Linux, há uma hierarquia onde a memória trabalha com preferência para processos que estão em execução. No momento em que um processo termina, se tiver espaço na memória, o sistema preservará resíduos desse processo na memória, para que a possibilidade de retorno ao mesmo processo seja mais rápida. No caso em que a memória RAM se encontre cheia com processos que estão em execução, a memória SWAP (virtual) será utilizada

como uma memória auxiliar para que o sistema não falhe, pois ela funciona com uma extensão da memória RAM.

2.1 Conceitos importantes

De acordo com Tanenbaum (2016), cada processo do Linux possui um espaço de endereçamento que compreende três segmentos: texto, dados e pilha.

O segmento de **texto** engloba as orientações da máquina que produzem o código executável do programa. Ele é gerado pelo compilador e montador que traduz, por exemplo C ou C++, em código de máquina. Outra característica do segmento de texto, em geral, é ser usado somente em leitura.

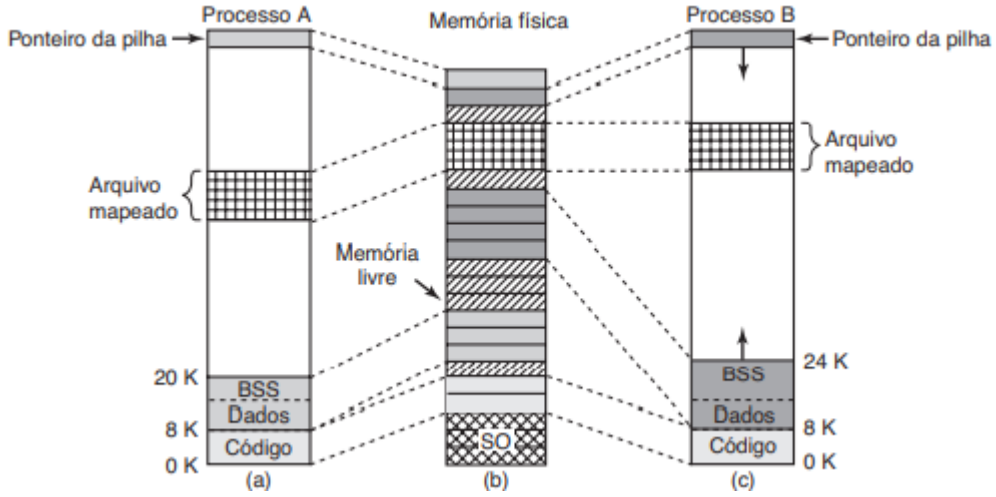
O segmento de **dados** envolve o armazenamento para todas as variáveis, vetores do programa, tal como outros dados. Esse segmento é dividido em duas partes, os dados inicializados e os não inicializados. A parte dos dados inicializados inclui variáveis e constantes do compilador que necessitam de um valor inicial quando o programa é inicializado. Já na parte dos dados não inicializados, todas as variáveis são inicializadas para zero logo depois do carregamento.

Como exemplifica Tanenbaum (2016), na linguagem C pode-se declarar uma cadeia de caracteres e inicia-la simultaneamente. No momento em que o programa é inicializado, ele aguarda que a cadeia tenha o seu valor inicial. A fim de efetuar essa construção, o compilador aponta à cadeia um local no espaço de endereçamento e garante que esse local comporte a cadeia de caracteres apropriada, quando o programa for inicializado. Na perspectiva do sistema operacional, os dados inicializados são um pouco semelhantes ao texto do programa, pois ambos possuem padrões de bits gerados pelo compilador que precisam ser carregados na memória quando o programa for inicializado.

O terceiro e último segmento é o de **pilha**. Em grande parte dos computadores, ele inicia perto do topo do espaço de endereçamento virtual e cresce para baixo na direção de 0, assim sendo, “Se a pilha cresce abaixo da parte de baixo do segmento de pilha, uma falta de hardware ocorre e o sistema operacional baixa a parte de baixo do segmento de pilha por uma página. ”

(TANENBAUM, 2016, p. 522). No momento em que um programa inicializa, sua pilha não está vazia. Ao invés disso, ela abarca todas as variáveis do ambiente, tal como a linha de comando digitada para o shell para chama-lo. Deste modo, um programa pode localizar seus argumentos.

Através desses três segmentos, os processos no Linux, além de alocar eficientemente mais memória, são capazes de obter dados de arquivos por meio de arquivos mapeados na memória. Essa qualidade torna possível mapear um arquivo para uma parcela do espaço de endereçamento do processo, de modo que o arquivo pode ser lido e escrito como se ele fosse um vetor de bytes na memória. Na imagem a seguir, é apresentado um arquivo que está mapeado em dois processos simultaneamente, em diferentes endereços virtuais.



Fonte: Tanenbaum (2016).

2.2 Implementação do gerenciamento de memória no Linux

Todos os processos do Linux, numa máquina de 32 bits, desfrutam de 3GB de espaço de endereçamento virtual para si próprio, enquanto o 1GB restante é reservado para suas tabelas de páginas e outros dados do núcleo. O 1GB do núcleo não é perceptivo quando o processo efetua no modo usuário, contudo fica disponível cada vez que o processo faz uma chamada ao núcleo. O espaço de endereçamento é gerado quando o processo é criado e sobrescrito em uma chamada ao sistema exec. Para permitir que diversos processos partirem a memória física subjacente, Tanenbaum (2016, p. 524) explica:

> O Linux monitora o uso da memória física, aloca mais memória conforme a necessidade dos processos do usuário ou componentes do núcleo, mapeia dinamicamente porções da memória física no espaço

> de endereçamento de diferentes processos, e dinamicamente traz para dentro e leva para fora da memória executáveis de programa, arquivos e outras informações de estado conforme a necessidade, de modo a utilizar os recursos da plataforma eficientemente e assegurar o progresso da execução.

2.3 Mecanismos de alocação de memória

O Linux oferece suporte a vários mecanismos para alocação da memória. O principal mecanismo para alocação de novos quadros de páginas de memória física é o alocador de páginas, que funciona ao utilizar o algoritmo companheiro (buddy algorithm). Segundo Tanenbaum (2016, p. 526):

> O Linux gerencia a memória usando o algoritmo companheiro, com a característica adicional de ter um arranjo no qual o primeiro elemento é o cabeçalho da lista de blocos do tamanho de 1 unidade, o segundo elemento é o cabeçalho de uma lista de blocos de tamanho de 2 unidades, o elemento seguinte aponta para blocos de 4 unidades, e assim por diante. Dessa maneira, qualquer bloco na potência de 2 pode ser encontrado rapidamente.

Entretanto, esse algoritmo conduz a uma fragmentação interna significativa, porque se um bloco de 65 páginas será usado, você deve pedir e receber um bloco de 128 páginas.

Para atenuar esse problema, o Linux também possui o alocador de fatias (slabs), que adquire blocos usando o algoritmo companheiro, porém corta fatias (unidades menores) a partir deles e gerencia as unidades menores de forma separada.

Há ainda um terceiro alocador de memória chamado vmalloc, que é utilizado no momento em que a memória solicitada precisa ser próxima somente no espaço virtual, mas não na memória física. Contudo, o uso de vmalloc ocasiona na diminuição de desempenho, e é usado essencialmente para alocar grandes quantidades de espaço de endereçamento virtual, como para inserir módulos de núcleo dinamicamente.

2.4 Representação do endereço virtual

O espaço de endereçamento virtual é dividido em áreas ou regiões homogêneas e alinhadas por páginas. De forma mais detalhada, cada área é “constituída por uma sequência de páginas consecutivas com a mesma proteção e propriedades de paginação. Pode haver brechas no espaço de endereçamento

virtual entre as áreas. Qualquer referência de memória a uma brecha resulta em uma falta de página fatal. ” (TANENBAUM, 2016, p. 526)

Todas as áreas são descritas no núcleo através de uma entrada vm_area_struct. Todos os vm_area_structs para um processo estão unidos em uma lista ordenada pelos endereços virtuais, de modo que todas as páginas podem ser achadas. Tanenbaum (2016, p. 527) ainda acrescenta:

> Quando a lista fica longa demais (mais de 32 entradas), é criada uma árvore para acelerar a sua busca. A entrada vm_area_struct lista as propriedades da área. Essas propriedades incluem o modo de proteção (por exemplo, somente leitura ou leitura/escrita), se ela está fixada na memória (não paginável), e em que direção ela cresce (para cima para segmentos de dados, para baixo para pilhas).

O vm_area_struct ainda aponta se a área é compartilhada com um ou mais processos ou é privada para o processo. Depoia de um fork, o Linux faz uma cópia da lista de área para o processo filho, porém caracteriza o pai e o filho para indicarem para as mesmas tabelas de páginas. Dessa forma, as áreas são indicadas como de leitura ou escrita, contudo as páginas em si são marcadas como somente de leitura.

2.5 Paginação no Linux

A unidade fundamental de gerenciamento de memória é uma página, e a maioria dos elementos de gerenciamento de memória atuam em uma granularidade de páginas. O pensamento por trás da paginação no Linux é simples: um processo não precisa estar inteiro na memória para ser executado. O que realmente é necessário é a estrutura do usuário e as tabelas de páginas. Quando elas são colocadas na memória, pode-se dizer que o processo é considerado “na memória” e está apto para ser escalonado e executado. Já as páginas do texto, dados e segmentos de pilha são trazidas, um de cada vez, conforme são referenciados. Tanenbaum (2016, p. 528) afirma:

> O Linux é um sistema que trabalha inteiramente por paginação por demanda, sem pré-paginação ou conceito de conjunto de trabalho (embora exista uma chamada na qual um usuário pode dar uma dica de que determinada página será necessária logo, na esperança de que ela esteja lá quando necessário). Segmentos de texto e arquivos mapeados são paginados para seus arquivos respectivos no disco. Tudo mais é paginado para a partição de paginação (se presente) ou um dos arquivos de paginação de comprimento fixo, chamados de área de troca.

Vale dizer que as páginas não são alocadas no dispositivo de paginação ou partição até o momento que elas sejam necessárias. Cada dispositivo inicia com um mapa de bits que informa quais páginas estão livres. Quando existe a necessidade de uma página sem armazenamento ser jogada para fora da memória, é selecionada a partição da paginação com maior prioridade ou arquivo que ainda tem espaço e uma página é alocada para ela.

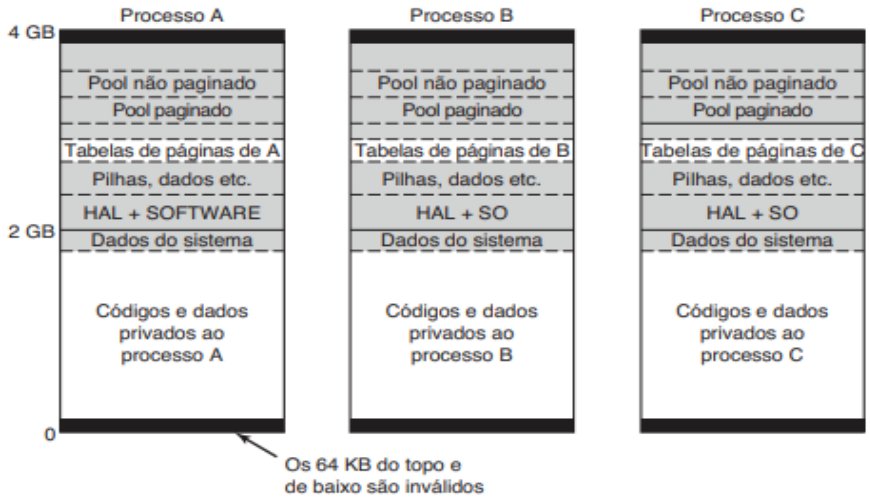
3 Gerenciamento de memória no Windows

O Windows utilizado como referência neste artigo é o Windows 8, caracterizado por ter um sistema de memória virtual extremamente sofisticado e complexo. Ele dispõe de diversas funções Win32 para usar a memória virtual, implementadas pelo gerenciador de memória.

3.1 **Conceitos importantes**

No Windows, cada processo de usuário possui seu próprio espaço de endereçamento virtual. Nas máquinas x86, os endereços virtuais possuem 32 bits de largura, logo, cada processo tem 4 GB de espaço de endereçamento virtual, com o usuário e o núcleo recebendo 2 GB cada. Já nas máquinas x64, o usuário e o núcleo recebem mais endereços virtuais do que eles poderiam razoavelmente utilizar no futuro previsto. Vale dizer que nas máquinas x86 e x64, o espaço de endereçamento virtual é paginado sob demanda.

A representação do espaço de endereçamento virtual para três processos x86 é mostrado é mostra abaixo de forma simplificada.



Fonte: Tanenbaum (2016).

Os 64KB do topo e da base do espaço de endereçamento virtual do processo geralmente não estão mapeados. Essa escolha foi proposital, buscando auxiliar o reconhecimento de erros de programação e diminuir a facilidade de exploração de certos tipos de vulnerabilidades.

Saindo dos 64 KB, vêm o código e os dados privados do usuário. Isso se estende por quase 2 GB. Os 2 GB superiores abarcam o sistema operacional, incluindo o código, dados e pools paginados e não paginados. Os 2 GB superiores constituem a memória virtual do núcleo, que é compartilhada entre todos os processos dos usuários, com exceção dos dados da memória virtual, como tabelas de páginas e listas do conjunto de trabalho, pois são exclusivas de cada processo. A memória virtual do núcleo somente está acessível quando em execução no modo núcleo. A razão para o compartilhar a memória virtual do processo com o núcleo é que, ao fazer uma chamada do sistema, o thread desloca o controle para o modo núcleo e continua executando sem mudar o mapa da memória. O Windows ainda permite que os threads se conectem a outros espaços de endereçamento no momento em que forem executados no modo núcleo.

3.1 Alocação de endereço virtual

As páginas de endereçamento virtual podem estar em um de três estados: inválida, reservada ou comprometida. Uma página **inválida** é caracterizada por não estar atualmente mapeada para um objeto de seção de memória, e uma referência a ela resulta em uma falta de página que ocasiona uma violação de acesso. Uma vez que o código ou os dados estejam mapeados em uma página virtual, diz-se que ela está **comprometida**. Uma falta de página em uma página comprometida sucede no mapeamento da página que contém a página virtual que ocasionou a falta em uma das representadas pelo objeto da seção ou armazenadas no arquivo de páginas.

Uma página virtual também pode estar no estado **reservada**. Uma página virtual reservada é inválida, mas com a característica de que os endereços virtuais nunca serão alocados pelo gerenciador de memória para nenhum outro propósito. Como Tanenbaum (2016, p. 645) exemplifica:

> Por exemplo, quando se cria um novo thread, são reservadas muitas páginas de espaço de pilha no espaço de endereçamento virtual do processo, mas somente uma fica comprometida. À medida que a pilha aumenta, o gerenciador de memória automaticamente compromete páginas adicionais até que a reserva seja quase exaurida. As páginas reservadas funcionam como páginas guardiãs, evitando que a pilha cresça demais e sobrescreva os dados de outros processos.

3.2 Arquivos de páginas

Em relação ao arquivo de páginas, “Há um engajamento interessante na atribuição da área de troca para as páginas comprometidas que não estejam sendo mapeadas para arquivos específicos. ” (TANENBAUM, 2016, p. 645). Porém, a questão é quando e como mapear a página virtual para uma localização específica dentro do arquivo.

Nesse caso, o Windows usa uma estratégia chamada just-in-time (no momento certo). As páginas comprometidas apoiadas pelo arquivo de páginas não recebem espaço nesse arquivo até que necessitem voltar para o disco. Se a memória virtual total é menor do que a memória física disponível, não há necessidade de um arquivo de páginas. O sistema também é inicializado dessa forma, já que os arquivos de páginas não são inicializados até que o primeiro processo no modo usuário, smss.exe, comece a executar.

Com a estratégia de pré-alocação, toda a memória virtual do sistema utilizada para o armazenamento de dados privados fica limitada ao tamanho do arquivo de páginas. Com o auxílio da alocação just-in-time, a memória virtual total pode ser tão grande quanto o tamanho dos arquivos de páginas e o da memória física combinados.

Com a paginação por demanda, as solicitações de leitura de páginas no disco devem ser imediatamente atendidas, porque o thread que encontrou a página ausente só pode prosseguir quando essa operação de entrada de página for concluída. Mas, as operações que escrevem as páginas alteradas no disco não costumam manter sincronia com a execução dos threads. Dessa forma, a estratégia just-in-time para alocação de espaço no arquivo de páginas toma proveito disso para aumentar o desempenho da operação de escrita de páginas modificadas no arquivo de páginas.

No momento em que as páginas armazenadas no arquivo de páginas são lidas para a memória, elas conservam sua alocação nesse arquivo até receberem a primeira modificação. Se uma página nunca for modificada, ela irá para uma lista especial de páginas físicas livres, chamada de lista de espera, onde pode ser utilizada novamente sem que precise ser escrita de volta no disco. Caso a página seja modificada, o gerenciador de memória irá liberar a página do arquivo de páginas e a cópia da página ficará na memória.

No final das contas, o sistema operacional é responsável por controlar a relação entre os mapas de páginas virtuais e o arquivo de páginas por meio da escrita dessa informação na entrada da tabela de páginas para o processo para páginas privadas ou nas entradas de protótipo da tabela de páginas associadas ao objeto da seção para páginas compartilhadas.

3.3 Paginação no Windows

No Windows, a pré-paginação é sustendada pelo gerenciador de memória, porém é implementada como um elemento separado do sistema. Isso é um ponto positivo para o sistema, já que as páginas levadas para a memória não são inseridas na tabela de páginas do processo, mas na lista de espera a partir da qual podem ser inseridas no processo rapidamente se for necessário, sem necessidade de acesso ao disco.

No caso das páginas não mapeadas, ocorre uma situação diferente, pois não são inicializadas na leitura do arquivo. Ao invés disso, a primeira vez que uma página não mapeada é acessada, o gerenciador de memória gera uma nova página física e certifica-se de que seu conteúdo seja somente zeros (por motivo de segurança).

A paginação por demanda no gerenciador de memória é produzida pelas faltas de página. A cada falta, há um desvio para o núcleo, que constrói um descritor independente de máquina advertindo o que aconteceu e passa esse descritor para a parte do executivo que opera o gerenciamento de memória. O gerenciador de memória irá verificar a validade do acesso. No caso de uma página compartilhada, o gerenciador de memória usa a entrada da tabela de páginas protótipo relacionada ao objeto de seção para poder preencher a nova entrada da tabela de páginas para a tabela de páginas do processo.

Quando o gerenciador de memória consegue atender uma falta de página encontrando a página necessária em vez de lê-la do disco, a falta é classificada como falta aparente. Se a cópia do disco for necessária, logo é uma falta estrita. Em comparação com as faltas estritas, as aparentes são mais baratas e causam menos impacto no desempenho da aplicação. Elas podem acontecer quando uma página compartilhada já foi mapeada em outro processo, quando somente uma nova página zerada é necessária ou quando a página solicitada foi eliminada do conjunto de trabalho do processo, porém é requisitada novamente antes de ter tido a chance de ser utilizada novamente.

Quando uma página física não está mais mapeada pela tabela de páginas de nenhum processo, ela é colocada em uma dessas listas: livre, modificada ou em espera. As páginas que não serão necessárias, como as de pilha de processos concluídos, são liberadas automaticamente. As que podem causar novas faltas vão para a lista de modificadas ou para a lista de espera. As páginas na lista de modificadas serão escritas no disco e então movidas para a lista de espera.

O gerenciador de memória ainda pode alocar páginas conforme necessário por meio da lista de livres ou da lista de espera. Antes de alocar uma página e copiá-la do disco, o gerenciador de memória irá verificar as listas de livres e de espera para confirmar se a página já está na memória. No Windows, o esquema de preparo transforma as futuras faltas estritas em faltas aparentes através do carregamento das páginas que devem ser necessárias e sua localização na lista de espera. O próprio gerenciador de memória faz uma pequena parte da pré-paginação acessando grupos de páginas consecutivas e não de páginas individuais.

3.4 Mudanças no Windows moderno

O Windows moderno adicionou uma camada de abstração extra no fundo do gerenciador de memória, chamada gerenciador de armazenamento. Essa camada é responsável por tomar decisões sobre como otimizar as operações de E/S ao armazenamento disponível. Os sistemas de armazenamento persistente incluem memória flash auxiliar e SSDs, além dos discos rotacionais. O gerenciador de armazenamento otimiza onde e como as páginas da memória

física são copiadas para o armazenamento persistente no sistema. Ele também implementa técnicas de otimização, como o compartilhamento do tipo copiar na escrita, para páginas físicas iguais, e compactação das páginas na lista de espera, para aumentar efetivamente a RAM disponível.

Outra importante mudança no gerenciamento de memória no Windows Moderno é a introdução de um arquivo de troca (swap). À medida que aumenta a pressão sobre a memória, o gerenciador de memória espreme os conjuntos de trabalho para reduzir as pegadas que cada processo deixa na memória. O modelo de aplicação moderno introduz oportunidades para novas eficiências. Visto que o processo que inclui a parte de primeiro plano de uma aplicação moderna não recebe mais recursos do processador depois que o usuário tiver saído dele, não é preciso que suas páginas fiquem residentes na memória. À medida que a pressão sobre a memória se acumula no sistema, as páginas no processo podem ser removidas como parte do gerenciamento normal do conjunto de trabalho. Entretanto, o gerenciador de tempo de vida do processo sabe por quanto tempo se passou desde que o usuário passou para o processo em primeiro plano da aplicação. Quando mais memória for necessária, ele seleciona um processo que não foi executado durante um tempo e convoca o gerenciador de memória para trocar todas as páginas em um pequeno número de operações de E/S. As páginas serão gravadas no arquivo de troca agregando-as em um ou mais trechos grandes. Isso significa que o processo inteiro também pode ser restaurado da memória com menos operações de E/S.

4. Considerações finais

A memória RAM é um item imprescindível para qualquer computador, pois quanto mais memória estiver instalada na máquina, melhor será seu desempenho. Porém não basta ter muita RAM disponível em seu computador, pois o que torna a RAM útil de verdade é a capacidade de uso que o Sistema Operacional faz dela através do gerenciamento de memória.

O gerenciamento de memória é o que torna o sistema operacional mais rápido e funcional. O gerenciador de memória deve ser capaz de controlar parte da memória está em uso (e quais não estão), alocar memória para processos quando eles necessitam e desalocar quando eles terminam e, especialmente,

gerenciar a troca entre a memória principal e o disco, quando a memória principal for pequena demais para armazenar todos os processos.

O Linux assim como o Windows, possui dois sistemas de gerenciamento de memória, o primeiro é a memória física, que cuida da alocação e liberação de blocos de memória, e o segundo é a memória virtual, que tem o papel de "driblar" os processos, informando que há mais memória do que realmente possui. Esta técnica aumenta a performance do sistema operacional.

Referências

TANENBAUM, A.S.; BOS, H. Sistemas Operacionais Modernos. 4. ed. São Paulo: Pearson Education do Brasil, 2016.