UNIVERSIDADE ESTADUAL DE LONDRINA

Guilherme Henrique Gonçalves Silva
Departamento de Computação
Universidade Estadual de Londrina Londrina, Brasil
guilherme.henriique.silva@uel.br

Gabriel Ângelo P. G. Sabaudo
Departamento de Computação
Universidade Estadual de Londrina Londrina, Brasil
gabriel.angelo@uel.br

Consistência em sistemas de arquivos

e

Desempenho do sistema de arquivos

Grupo 7 - Gabriel Sabaudo e Guilherme Henrique

Londrina 2021

Resumo

Muitos sistemas de arquivos leem os blocos, modificam seu conteúdo e só depois os escrevem. Se o sistema cair antes que todos os blocos alterados tenham sido escritos, o sistema de arquivos poderá ficar em um estado inconsistente. Esse problema será especialmente crítico se alguns dos blocos ainda não escritos forem de i-node, blocos de diretório ou blocos que contenham a lista de blocos livres.

O acesso à arquivos é parte vital da estrutura de organização de dados de um sistema, porém esse acesso pode ser prejudicado devido a diferença do tempo de acesso em discos rígidos e memórias. Por este fator, muitos sistemas de arquivos foram implementados com algumas técnicas que melhoram o desempenho de acesso, sendo elas o Cache de blocos, Leitura antecipada de blocos e a Redução do movimento do braço de disco rígido.

abstract

Many file systems read the blocks, modify their contents, and then write them. If the system goes down before all changed blocks have been written, the file system can be left in an inconsistent state. This problem will be especially critical if some of the blocks not yet written are i-node, directory blocks or blocks that contain the list of free blocks.

File access is a vital part of a system's data organization structure, but this access can be hampered due to the difference in access time on hard disks and memories. For this reason, many file systems have been implemented with some techniques that improve access performance, such as Block Cache, Block Read Ahead and Hard Disk Arm Movement Reduction.

Consistência do sistema de arquivos

Como você provavelmente já sabe, tudo é considerado um arquivo no Linux. Isso inclui dispositivos de hardware, processos, diretórios, arquivos regulares, soquetes, links e assim por diante. Além disso, geralmente, o sistema de arquivos se divide em blocos de dados e i-nodes. Com isso dito, você pode pensar nos i-nodes como a base do sistema de arquivos Linux. Para explicar mais claramente, um i-node é uma estrutura de dados que armazena metadados sobre cada arquivo em seu sistema de PC.

Para tratar do problema de inconsistência nos sistemas de arquivos, a maioria dos computadores tem um programa utilitário que verifica a consistência do sistema de arquivos. Por exemplo, o UNIX tem o fsck e o Windows tem o scandisk. Esse utilitário pode ser executado sempre que o sistema estiver sendo inicializado, especialmente depois de uma queda.

O princípio geral de usar a redundância inerente do sistema de arquivos para consertá-lo vale para os dois. Todos os verificadores conferem cada sistema de arquivos (participação do disco) independentemente dos outros.

Existem dois tipos de verificações de consistência: por blocos e por arquivos. Para verificar a consistência dos blocos, o programa constrói duas tabelas, cada uma com um contador para cada bloco, inicialmente contendo 0. Os contadores da primeira tabela monitoram quantas vezes cada bloco está presente em um arquivo; os contadores da segunda tabela registram quantas vezes cada bloco está presente na lista de livres (ou mapas de bits de bloco livres).

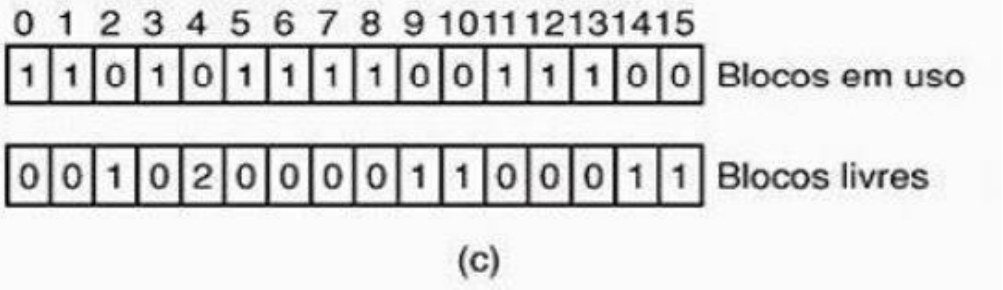
O programa, então, lê todos os i-nodes usando um dispositivo cru (raw), isto é, que ignora a estrutura dos arquivos e apenas devolve todos os blocos do disco começando no 0. A partir de um i-node, é possível construir uma lista de todos os números de blocos usados no arquivo correspondentes, conforme cada número de bloco é lido seu contador na primeira tabela é incrementado. O programa então verifica a lista ou o mapa de bits de livres para encontrar todos os blocos que não estiverem sendo usados. Cada ocorrência de um bloco na lista de livres faz com que seu contador na segunda tabela seja incrementado.

Se o sistema de arquivos estiver inconsistente, cada bloco terá um 1 ou na primeira ou na segunda tabela, como mostra a Figura A.

Contudo, depois de uma queda no sistema, as tabelas podem ficar como mostrado na Figura B na qual o bloco 2 não ocorre em nenhuma tabela. Será reportado como um bloco desaparecido. Embora os blocos desaparecidos não causem nenhum mal, eles ocupam espaço e, portanto, reduzem a capacidade do disco. A solução para os blocos desaparecidos é simples: o verificador de sistema de arquivos apenas os inclui na lista de blocos livres.

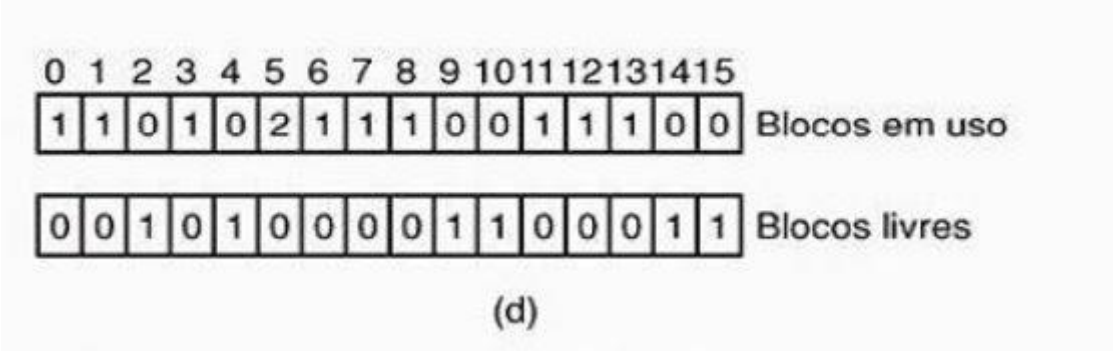
(b)

Outra situação possível é a mostrada na Figura C. Nela vemos um bloco, o número 4, que ocorre duas vezes na lista de livres. A solução aqui também é bem simples: reconstruir a lista de livres.



(c)

A pior coisa que pode ocorrer é o mesmo bloco de dados estar presente em dois ou mais arquivos, como mostra a figura D com o bloco 5. Se um desses arquivos for removido, o bloco 5 será colocado na lista de livres, resultando na situação em que o mesmo bloco está em uso e livre ao mesmo tempo. Se ambos os arquivos forem removidos, o bloco será colocado na lista de livres duas vezes. O que o verificador de sistema de arquivos deve fazer é alocar um bloco livre, copiar o conteúdo do bloco 5 nele e inserir a cópia em um dos arquivos. Desse modo o conteúdo dos arquivos permanece inalterado, mas a estrutura do sistema de arquivos, pelo menos, fica consistente. O erro deve ser reportado para permitir que o usuário impressione o dano.



(d)

Além de verificar se cada bloco está contabilizando corretamente, o verificador de sistemas de arquivos também checa os sistemas de diretórios. Além disso, ele usa uma tabela de contadores, mas por arquivos, e não por blocos. Ele inicializa a partir do diretório raiz e recursivamente percorre a árvore, inspecionando cada diretório do sistema de arquivos. Para cada arquivo no diretório, ele incrementa um contador para contar o uso de arquivos. Lembre-se de que, por causa de ligações estritas (hard links), um arquivo pode aparecer em 2 ou mais diretórios. As ligações simbólicas não contam e não fazem com que o contador incremente para o arquivo alvo.

Quando estiver tudo terminado, haverá uma lista, indexada pelo número de i-node, indicando quantos diretórios cada arquivo contém. Eles então compararam esses números com a contagem de ligações armazenadas nos próprios i-nodes. Essas contagens iniciam em 1 quando um arquivo é criado e são implementadas a cada vez que uma ligação estrita é feita para o arquivo. Em um sistema de arquivos consistente, os computadores devem ser iguais. Contudo, há a possibilidade de ocorrer 2 tipos de erros: a contagem de ligações no i-node pode ser alta ou baixa demais.

Se a contagem de ligações for mais alta que o número de entradas de diretório, então, mesmo que todos os arquivos sejam removidos dos diretórios, a contagem ainda será diferente de zero e o i-node não será removido. Esse erro não é grave, mas consome espaço de disco com os arquivos que não ocupam nenhum diretório. Ele deve ser reparado atribuindo-se o valor correto à contagem de ligações no i-node.

O outro erro é potencialmente catastrófico. Se duas entradas de diretório foram redirecionadas para um arquivo, mas o i-node indicar que há somente uma, quando uma das entradas de diretório for removida, a contagem do i-note irá para zero. Quando uma contagem do i-node irá para zero, o sistema de arquivos marca como não usado e libera todos os seus blocos. Essa ação resultará em um dos diretores apontando agora para um i-node não usado, cujo blocos podem logo ser atribuídos a outros arquivos. Novamente, a solução é forçar a contagem de ligações a assumir o número real de entradas no diretório.

Desempenho do Sistema de Arquivos

Pelo fato de que o acesso ao disco rígido leva muito mais tempo do que o acesso à memória, foram criadas algumas técnicas que melhoram o desempenho no sistema de arquivos.
Algumas delas são:

- Cache de blocos
- Leitura antecipada de blocos
- Redução do movimento do braço de disco

1. Cache de Blocos

É a técnica mais utilizada para fins de desempenho no sistema de arquivos. Possui muitos algoritmos que lidam com o gerenciamento da cache em disco, sendo o mais

comum responsável por verificar todas as requisições de leitura de modo a identificar se o bloco desejado se encontra na cache.

Caso o bloco esteja na cache, a requisição de leitura pode ser atendida sem um acesso ao disco (HIT).

Caso o bloco não esteja na cache, ele primeiro será lido do disco e repassado ao processo requisitante. Subsequentemente ele será copiado para a cache.

2. Leitura antecipada de blocos

Busca transferir blocos para a cache de disco antes que eles sejam necessários. Essa previsão tem como objetivo aumentar a taxa de acertos, e devido a fatores históricos, muitos programas acessam arquivos de maneira sequencial para que este método possa ser implementado.

Quando um bloco k é solicitado ao sistema de arquivos ele o busca e verifica se o bloco k+1 já se encontra na cache; Em caso negativo, ele agenda para recuperação.

Para acesso aleatório, esta estratégia piora o tempo de acesso, pois a imprevisibilidade aumenta de forma considerável, sendo necessário algoritmos que possam monitorar o comportamento dos acessos e assim poder “arriscar” o acesso.

3. Redução do movimento do braço de disco

Tem como objetivo reduzir o tempo do movimento do braço do disco rígido, considerando que o disco gira a uma velocidade muito maior do que o braço pode acompanhar.

Minimizar o número de movimentos do braço para efetuar a leitura dos blocos de um arquivo pode otimizar em muito o desempenho do sistema de arquivos, e pode ser alcançado via uma reorganização inteligente dos blocos dos arquivos no disco.

Referências

Tanenbaum, Andrew S. SISTEMAS OPERACIONAIS MODERNOS: 3° Edição. São Paulo: Pearson Education do Brasil, 2010