QUESTIONÁRIO – SISTEMA DE ARQUIVOS T2FS

Vinicius Amaro Cechin 159984;Douglas Cavalli Rachevsky 205686; Giuseppe Moroni Ramella 208781 208781

1. Sem alterar a quantidade de ponteiros de alocação indexada do Superbloco, quais outros fatores influenciam no maior tamanho de arquivo T2FS possível? Como esse fatores influenciam nessa tamanho?
O tamanho do BitMap, pois ele limita o número de blocos que pode-se controlar a alocação, e o tamanho da area de dados do disco, que limita fisicamente a quantidade de informações que podem ser armazenadas. Estes fatores também determinam o tamanho máximo do arquivo.

2. Supondo que você desejasse melhorar o T2FS, permitindo a criação de vínculos estritos (hardlinks). Que alterações seriam necessárias no T2FS? Há necessidade da criação de novas funções? Se sim, quais? Se não, porque não.
Seria necessário criar contadores de hardlinks para cada arquivo pois não pode-se deletar um arquivo que tenha mais de 1 descritor apontando para ele, para implementar o hardlink em si, basta criar outro descritor com os mesmos ponteiros que o descritor principal do arquivo. A função de delete deveria checar se o contador está em 1 antes de deletar o arquivo, e a função write deveria atualizar os ponteiros de todos os descritores que apontam pro arquivo. É necessário criar uma função apenas para criar novo hardlink.

3. As estruturas de controle do T2FS contêm informações que permitem verificar a consistência de alguns de seus elementos. Isso é possível graças a um certo nível de redundância de informação (por exemplo, no registro de arquivo, nas entradas do diretório, o número total de blocos usados por um arquivo e o tamanho do arquivo – em bytes – permitem uma verificação). Identifique quais outros elementos são redundantes e discuta como seria possível usar essa redundância para aumentar a confiabilidade do T2FS.
É possível também recuperar o tamanho do arquivo pelo numero de ponteiros que não estão em 0. É possível utilizar essa redundância para detectar falhas e estruturas inconsistentes no sistema de arquivos. Se uma das formas de verificar o tamanho do arquivo está diferente de outra, isso provavelmente ocorreu por uma queda de sistema, causado por exemplo por uma queda de energia. Com estas informações, é possível definir rotinas para checar qual a informação correta e recuperar os dados.

4. Como você implementou a atribuição dos identificadores de arquivos (file handler) pelas funções t2fs_create e t2fs_open? Discuta a questão da reutilização dos mesmos.
Os identificadores de arquivos são gerados a partir de uma variavel global que inicia em 0 e incrementa a cada create/open, retornando um identificador único e adicionando-o a uma lista que relaciona o descritor do arquivo, com o identificador e a posição do cursor. A reutilização do identificador pode acontecer sem erros caso o arquivo já tenha sido fechado ou deletado, pois nessas funções o identificador pode ser removido da lista de controle dos arquivos tornando-se inválido.

5. Como você implementou a gerência do contador de posição (current pointer) usado pela função t2fs_seek?
Quando um arquivo é aberto ou criado, é adicionado um elemento em uma lista do sistema de arquivos que relaciona o identificador do arquivo, a posição do descritor e a posição do cursor, que inicia em 0. A função t2fs_seek vai apenas atualizar a posição do cursor procurando na lista pelo identificador passado como parâmetro para a função. Sempre que é necessário a informação da posição do cursor, uma função é chamada com o parametro do identificador de arquivo, que varre a lista e retorna a posição do cursor.

6. A escrita em um arquivo (realizada pela função t2fs_write) requer uma sequência de leituras e escritas de blocos de dados e de blocos de controle. Qual é a sequência usada por essa função? Se essa sequência for interrompida (por falta de energia, por exemplo) entre duas operações de escrita de bloco, qual será o efeito na consistência dos dados no disco? É possível projetar uma sequência de escritas no disco que minimize a eventual perda de dados?
Na escrita, é feito primeiro a escrita nos blocos de dados, depois realiza a escrita nos blocos de índice e nos handles de arquivo, e por fim escreve no bitmap quais blocos foram alocados (se foi necessário alocar mais algum para escrita de todos os dados). Esta ordem foi escolhida para que o usuário só enxergar que existem dados escritos quando efetivamente eles já estão no disco. O efeito negativo desta implementação é que os blocos de disco serão alocados tão logo forem escritos, e, caso ocorra uma falta de energia, estes blocos não serão apontados como alocados pelo bitmap mesmo que estes blocos estejam com dados pertencentes ao arquivo que foi escrito. O que pode causar uma perda de dados, pois os blocos alocados onde foram escritos os dados, não foram escritos no bitmap. Assim o sistema trata aquele bloco como não tendo dados, e um possível bloco para escrita de outro arquivo, o que causa dois arquivos apontando para o mesmo bloco. Escrevendo-se o bitmap primeiro, em seguida os blocos de indice, e por fim os blocos de dados, diminui a perda de dados, mas gera outros problemas, como alocação de blocos não utilizados.

7. Algumas estruturas gravadas no disco são mais facilmente manipuláveis se estiverem na memória principal (como se fosse uma cachê). Por outro lado, isso aumenta a possibilidade de perda de dados, pois as informações existentes nessa cachê e que não foram escritas no disco, podem ser perdidas, caso ocorra alguma interrupção de operação do sistema. Quais informações do disco você está mantendo (e gerenciando) na memória principal e porque você as escolheu? Qual a política que você usou para decidir quando escrevê-las no disco?
Optei por gravar todas as informações que são armazenadas no disco na hora em que são alteradas, ou no fim da função que a alocou (como é o caso das variaveis globais: DirAtual, bitmap e superbloco). A única informação que é mantida em memória é a lista de identificadores de arquivos, juntamente com os apontadores de posição do cursor de cada arquivo aberto. Apesar de perder em eficiência, garante uma melhor recuperação de dados.

8. Todas as funções implementadas funcionam corretamente? Relate, para cada uma das funções desenvolvidas, como elas foram testadas?
Sim, todas as funções propostas foram implementadas e funcionam como deveria (pelo menos para os casos testados). Elas foram testadas sempre com uma imagem de disco nova, testando varias possibilidades de retornos das mesmas.

9. Relate as suas maiores dificuldades no desenvolvimento deste trabalho e como elas foram contornadas
As maiores dificuldades foram a implementação da função create e delete, ambas envolviam alterações em varias partes do disco, o que gerou muitas dificuldades para descobrir como superar esses problemas. Apareceram também questões mais simples de implementação que foram resolvidas mais facilmente. No geral, implementar todas as funções foi trabalhoso, mas sua compreensão tornou-se mais fácil ao longo do processo de implementação.