Relatorio trabalho 2 de Sistemas Operacionais

Miller Biazus e Pedro Morales

1. Sem alterar a quantidade de ponteiros de alocação indexada, quais outros fatores influenciam no maior tamanho de arquivo T2FS possível? Como esses fatores influenciam nesse tamanho?
Basicamente, o tamanho dos blocos (numero de bytes) e espaço usado para armazenar o ponteiro de um bloco (tamanho do indice de identificação de um bloco no disco. Se o tamanho usado para os ponteiros dos blocos for menor, sobra mais espaço no disco para alocar bytes de um arquivo. Em contrapartida se um bloco tiver maior tamanho, proporcionalmente caberão arquivos maiores relativos a este acréscimo de tamanho.

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.
Criaria uma função create_HL(char* name, t2fs_file file), que receberia como parametro o nome do hardlink e o arquivo para o qual o hardlink servirá de atalho. Creio que a proxima alteração necessária seria criar um contador de arquivos. Ou seja, isto serviria para controle(se o arquivo no disco deveria ser deletado ou não após a deleção de um hardlink, por exemplo).

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 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.
Vetor de bitmaps é redundante. Afinal um bloco poderia ser verificado se está vazio ou não a partir dos ponteiros para os blocos de dados. Assim pode-se fazer uma checagem para assegurar que as informações do bitmap está correta, aumentando a confiabilidade do programa.

4. Como você implementou a gerência do contador de posição (current pointer) usado pela função seek2?

A estrutura handlers recebe o nome do arquivo em questão e o ponteiro onde a “ação” está sendo feita. Então no caso de sucesso na criação de um arquivo o currentPointer é retornado

com o valor 0, e quando um arquivo é fechado seu handler também é finalizado, e o vetor de handlers atualizado (o handler atual é removido para que outro possa ser colocado lá).

5. A escrita em um arquivo (realizada pela função write2) 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?

O registro que mantem as informacoes de controle do arquivo apontado por handle é procurado na estrutura de diretórios, verifica-se em qual posicao do arquivo o currentPointer está; caso seja no meio do arquivo, um bloco de dados que ja foi alocado previamente é lido do disco, caso o currentPointer esteja apos o final do arquivo é necessario verficar a necessidade de se alocar um novo bloco de dados, o que resulta em um acesso do bitmap no disco e sua modificacao, para entao realizar a escrita no disco dos dados armazenados em buffer. O fato de ocorrer uma nova alocacao implica em uma eventual atualizacao do registro do arquivo no disco, caso precisou-se criar um novo ponteiro direto ou indireto, e tambem para armazenar a modificacao do tamanho em bytes e em blocos do arquivo. Caso ocorra uma interrupcao no meio desse processo, pode ocorrer de o arquivo apresentar mais bytes escritos do que seu registro aponta, o bitmap pode estar desatualizado, mostrando como livres os blocos que foram escritos na porcao da funcao que executou antes da interrupcao. Ao se modificar as estruturas de controle somente após a escrita do arquivo se evita que registros inválidos interfiram com a manipulacao dos outros arquivos, assim prejudicando somente o arquivo que teve sua escrita interrompida.

6. Algumas estruturas gravadas no disco são mais facilmente manipuláveis se estiverem na memória principal (como se fosse uma cache). Por outro lado, isso aumenta a possibilidade de perda de dados, pois as informações existentes nessa cache 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?

A estrutura do superbloco (informações do superbloco) é mantida em memória, já que não há a necessidade de escrevê-la no disco novamente.
O buffer de um determinado bloco lido é mantido em memória enquanto o manipulamos. O write só é feito depois da manipulação completa do bloco, ou seja, só quando há certeza de que a manipulação do bloco já foi executada.

7. Todas as funções implementadas funcionam corretamente? Relate, para cada uma das funções desenvolvidas, como elas foram testadas?

As funções foram testadas ao longo do desenvolvimento, já que algumas dependiam de outras (por exemplo a função create dependia de manipulação de strings para o path e também de funções que percorriam os vários níveis do path procurando local livre no diretório pai do arquivo). Fomos desenvolvendo todas as funções em paralelo de acordo com o que tínhamos feito, Começamos estruturando o projeto (criando funções de manipulação de blocos, bitmap, indices etc.) e isso nos deixou com pouco tempo apra terminar todas as funções.

8. Relate as suas maiores dificuldades no desenvolvimento deste trabalho e como elas foram contornadas.

Tempo e projeto. O projeto é muito grande e para se obter sucesso é necessário mais tempo. É dificil conciliar tanta coisa, já que o trabalho é lançado quando ainda não se tem muita noção de arquivos. Uma sugestão é que o trabalho fosse solicitado em incrementos, pois a gente passou mais tempo tentando estruturar o T2FS do que o programando em si.