O Sistema de Arquivos Unix File System

Trabalho 3 da disciplina Sistemas Operacionais I

Gustavo Emanuel Kundlatsch
Departamento de Informática e Estatística
Universidade Federal de Santa Catarina
Florianópolis, Brasil
gustavo.kundlatsch@gmail.com

Gustavo Raimundo
Departamento de Informática e Estatística
Universidade Federal de Santa Catarina
Florianópolis, Brasil
incio.gusta@gmail.com

***Resumo*—Esse trabalho foi desenvolvido para a matéria INE5412, no primeiro semestre do ano letivo de 2019. Nele, foram estudados os princípios teóricos e práticos do sistema de arquivos *Unix File System* (UFS). Esse sistema de arquivos foi criado para suprir as demandas que surgiram com o avanço da tecnologia, que tornaram defasados os princípios do sistema de arquivos tradicional do Unix.**

***Palavras Chave*—file system, unix, UFS**

I. Introdução

Para compreender o funcionamento do UFS (que também é chamado de Berkeley Fast File System, BSD Fast File System ou Fast File System), precisamos compreender como era o sistema de arquivos de versões mais antigos do Unix. O sistema de arquivos original, descrito no clássico artigo de Thompson que fazia a descrição do funcionamento do Unix [1], era chamado apenas de FS, e era formado apenas por um bloco de inicialização, superbloco, um conjunto de inodes e os blocos de dados. Nos primórdios do Unix, esse sistema de arquivos era efetivo apenas por conta da baixa capacidade dos discos, quando não precisava de métodos muito robustos para garantir eficiência. Com o passar do tempo, a tecnologia avançou e a capacidade dos discos aumentou, e os movimentos dos cabeçotes para se deslocar através os inodes e os blocos de dados acabaram sendo caracterizados como causadores de problemas de performance [2], num processo conhecido como Thrashing - situação onde muito recurso computacional é utilizado para realizar uma quantidade mínima de trabalho.

O UFS trouxe conceitos novos com o objetivo de melhorar o desempenho e impedir o Thrashing, com a criação de grupos de cilindros e a divisão do disco em pedaços menores, cada um com seu próprio conjunto de inodes e blocos de dados. O UFS é uma reimplementação do antigo FS, e fornece taxas de transferência consideravelmente superiores usando políticas de alocação mais flexíveis que permitem uma melhor localização de referência e podem ser adaptadas a uma ampla gama de características do processador. Esse sistema de arquivos foi proposto por McKusick et. al. [3], e sua implementação original teve taxas de acesso a arquivos de até dez vezes mais rápido que o sistema de arquivos UNIX tradicional.

II. O Sistema de Arquivos UFS

Nessa seção, iremos discutir como funciona o UFS, de acordo com foi descrito por McKusick et. al. [3] em 1984.

A. Conceitos Básicos

Na organização do sistema UFS, cada partição do disco contém um ou mais sistemas de arquivos, diferente do antigo FS. Um sistema de arquivos é descrito por seu superbloco, localizado no início da partição. Como o super-bloco contém dados críticos, ele é replicado para proteger eventuais perdas. Isso é feito quando o sistema de arquivos é criado; uma vez que os dados do super-bloco não mudam, as cópias não precisam ser referenciadas, a menos que uma falha no cabeçalho ou outro erro no disco rígido cause algum problema permanente mais severo.

A organização do sistema UFS divide uma partição de disco em uma ou mais áreas chamadas *cylinder groups*, ou grupo de cilindros. Um grupo de cilindros é composto por um ou mais cilindros, que são consecutivos em um disco. Associado a cada grupo de cilindros estão algumas informações que incluem a cópia redundante do super-bloco, espaço para inodes, um mapa de bits que descreve os blocos disponíveis no grupo de cilindros e informações resumidas descrevendo o uso de blocos de dados dentro do grupo de cilindros. Os super-blocos armazenam informações sobre um sistema de arquivos específico, portanto é importante guardar uma cópia para recuperar essas informações caso aconteça algum imprevisto. Os inodes possuem todas as informação necessária para que o sistema de arquivos possa manipular os arquivos e diretórios, e cada arquivo é representado por um inode no sistema de arquivos. O mapa de bits dos blocos disponíveis no grupo de cilindros substitui a lista encadeada de endereços livres do sistema de arquivos anterior do Unix. Para cada grupo de cilindros, um número estático de inodes é alocado no tempo de criação do sistema de arquivos. A política padrão é alocar um inode para cada 2048 bytes de espaço no grupo de cilindros, esperando que isso seja muito mais do que o necessário e que nunca será necessário aumentar o tamanho.

Todas as informações do grupo de cilindros podem ser colocadas no início do mesmo. No entanto, se isso fosse feito assim, todas as informações redundantes estariam no começo. Para evitar problemas, a informação do grupo de cilindros

começa em um deslocamento variável a partir do início do grupo de cilindros. O deslocamento para cada grupo de cilindros sucessivos é calculado em cerca de uma pista a partir do início do grupo de cilindros do que o grupo de cilindros anterior. Dessa maneira, as cópias de segurança redundantes espiralam ao redor do disco, de modo que qualquer cilindro ou prato possa ser perdido sem perder todas as cópias de superbloco.

B. Design

Um volume é a parte do disco que você interage como usuário [4]. No sistema UFS, um volume é composto dos seguintes elementos [5]:

- Blocos reservados para blocos de boot, no inicio da partição, que são inicializados separadamente do sistema de arquivos;
- Um superbloco, que contem um número mágico (uma constante especial utilizada como identificador) correspondente a um sistema de arquivos UFS, e números que descrevem sua geometria, estatísticas e parâmetros para ajuste comportamental;
- Um conjunto de grupos de cilindros. Cada grupo contém os seguintes componentes:
  - Um backup do superbloco;
  - Um cabeçalho de grupo de cilindros que contém diversas informações sobre o grupo de cilindros, parecido com o do superbloco;
  - Uma quantidade de inodes, cada um contendo atributos de arquivos;
  - Um número de blocos de dados;

Os blocos reservados para blocos de boot são utilizados ao incializar a máquina, quando a partição ativa executa esses blocos (também chamados de blocos de inicialização) para que o SO seja carregado em memória. Os arquivos de diretório contêm apenas a lista de nomes de arquivos no diretório e o inode associado a cada arquivo. Todos os metadados de arquivo (como criador, proprietário atual, data de criação, tamanho máximo, etc) são mantidos no inode.

C. Otimização do uso de armazenamento

No UFS, como descrito por McKusick et. al. [3], os dados são dispostos de modo que os blocos maiores possam ser transferidos em uma única transação de disco, tendo como objetivo aumentar a taxa de transferência de arquivos. Por exemplo, considerando um arquivo no sistema UFS composto por 4096 bytes de dados, no sistema anterior de arquivos do Unix esse arquivo seria composto de blocos de 1024 bytes. Vale lembrar que blocos tem tamanho de armazenamento fixo. Aumentando o tamanho do bloco, os acessos ao disco no sistema UFS podem transferir até quatro vezes mais informações por transação de disco. Em arquivos grandes, vários blocos de 4096 bytes podem ser alocados a partir do mesmo cilindro, de modo que até mesmo transferências de dados maiores sejam possíveis antes de exigir uma busca.

O ponto negativo de usar blocos maiores é a ineficiência ao desperdiçar armazenamento quando se guardam muitos arquivos de tamanho pequeno, que ocuparão um bloco completo de 4096 bytes. E o problema é que a maioria dos sistemas de arquivos Unix é composta de muitos arquivos pequenos.

Para conseguir aproveitar a vantagem do uso de blocos maiores, mas sem desperdiçar tanta memória, arquivos pequenos precisam ser armazenados de maneira mais adequada. O sistema UFS permite a divisão de um único bloco do sistema de arquivos em um ou mais fragmentos. O tamanho do fragmento do sistema de arquivos é especificado no momento em que o sistema de arquivos é criado, cada um desses blocos pode ser dividido em 2, 4 ou 8 fragmentos. Há um limite inferior do tamanho desses fragmentos, para evitar que haja uma granulação muito elevada, com fragmentos muito pequenos. Isso é restrito pelo tamanho do setor do disco, normalmente 512 bytes. O mapa de bits dos blocos associado a cada grupo de cilindros registra o espaço disponível em um grupo de cilindros no nível do fragmento. Para determinar se um bloco está disponível, os fragmentos alinhados são examinados.

D. Parametrização do sistema de arquivos

McKusick et. al. [3] propões em seu artigo que o sistema de arquivos receba parâmetros como entrada para a sua configuração, para que ele possa de adaptar de acordo com o hardware do computador.

Exceto para a instanciação da lista encadeada de blocos livres, o sistema de arquivos tradicional do Unix ignora as diferenças de hardware que cada dispositivo possuí. O FS não possuía qualquer informação a respeito do computador ou do disco que estava operando. No UFS, as características do processador e as características do disco são levadas em conta, de modo que os blocos podem ser alocados de uma maneira otimizada de acordo com a configuração. As variáveis que são consideradas incluem a velocidade do processador, o suporte de hardware para transferências de dados em massa e as características dos dispositivos de armazenamento. A tecnologia de disco está em constante aprimoramento, e a instalação pode ter várias tecnologias de disco diferentes sendo executadas em um único processador (hoje em dia é bastante comum notebooks terem um HD para o armazenamento de dados e um SSD para o SO, para otimizar o boot, por exemplo). Cada sistema de arquivos é parametrizado para que possa ser adaptado às características do disco em que é colocado.

Dessa maneira, a velocidade de transmissão de dados para o disco rígido é melhorada de maneira genérica, independente da máquina que está sendo utilizada, da quantidade de discos ou da tecnologia utilizada pelos discos. Esse aprimoramento pode ser aplicado em qualquer contexto, inclusive com dispositivos de armazenamento que possuem características distintas.

Essas características físicas de cada disco consideradas incluem o número de blocos por faixa e a taxa de rotação do disco. Quando os braços vão ser movimentados, as rotinas de alocação usam essas informações para calcular o

número de milissegundos necessários para pular um bloco. Quando um bloco é alocado para um arquivo, as rotinas de alocação calculam o número de blocos a serem pulados para que o próximo bloco no arquivo fique em posição seguinte do registro na quantidade esperada de tempo que leva para iniciar uma nova operação de transferência de disco. Para programas que acessam sequencialmente grandes quantidades de dados, essa estratégia minimiza a quantidade de tempo gasto esperando pelo disco para se posicionar, aproveitando a localidade espacial.

Outra maneira de otimizar o armazenamento de arquivos é ter uma boa política de alocação. O UFS tenta alocar novos blocos no mesmo cilindro que o bloco anterior do mesmo arquivo. O ideal é que esses novos blocos também estejam rotativamente bem posicionados, para acompanhar a rotação do disco sem precisar reposicionar o braço. A distância entre os blocos posicionados de maneira otimizada pode variar bastante. Em um processador com um canal de entrada/saída que não requer qualquer intervenção do processador entre solicitações de armazenamento (utilizando um dispositivo DMA, por exemplo), dois blocos de disco consecutivos podem ser acessados sem sofrer perda de tempo devido a necessidade de uma rotação de disco completa. Para processadores sem canais de entrada/saída, existe um custo de processamento a ser pago, uma vez que o processador precisa controlar essa transferência de dados.

E. Polítcas de layout

No UFS, as políticas de layout são divididas em dois níveis, o nível superior e o nível inferior. No nível superior se encontram políticas globais, que fazem uso de informações resumidas do sistema de arquivos para realizar o posicionamento dos blocos de dados novos e dos inodes. São essas rotinas as responsáveis por tomar a decisão de onde posicionar novos arquivos. No nível inferior, abaixo das rotinas de políticas globais, ficam as rotinas locais de alocação, que usam um esquema do alocação otimizada para o layout do bloco de dados.

As políticas globais de layout tentam melhorar o desempenho ao agrupar as informações relacionadas. Para evitar um sobrecarregamento do sistema, não é possível localizar todas as referências de dados para agrupar de maneira perfeita, pois isso teria um custo computacional muito elevado. Por conta disso, as políticas globais tentam distribuir dados não relacionados entre diferentes grupos de cilindros. Se houver muita tentativa de localização, o grupo de cilindros local poderá ficar sem espaço forçando os dados a serem espalhados para grupos de cilindros não locais.

A política de layouts são responsáveis também por alocar os inodes. Inodes são usados para descrever arquivos e diretórios, guardando atributos, como já foi descrito anteriormente, e os inodes de arquivos no mesmo diretório são frequentemente acessados juntos. Por exemplo, o comando "ls" do linux, que mostra a lista de arquivos em um diretório, normalmente acessa o inode de cada um dos arquivo do respectivo diretório. A política de layout tenta colocar todos os inodes de arquivos que estão em um mesmo diretório no mesmo grupo de cilindros. Para garantir que os arquivos sejam distribuídos por todo o disco, uma política diferente é usada para a alocação de diretórios. Um novo diretório é colocado em um grupo de cilindros que possui um número de inodes livres maior que a média, e o menor número de diretórios que já estão nele. A alocação de inodes dentro do grupo de cilindros é feita usando uma estratégia de "próximo livre". Embora isso aloque os inodes aleatoriamente dentro do grupo de cilindros, todos os inodes de um determinado grupo de cilindros podem ser lidos com 8 a 16 transferências de disco. (No máximo 16 transferências de disco são necessárias porque um grupo de cilindros pode ter no máximo 2048 inodes). Isso fornece um limite superior pequeno e constante no número de transferências de disco necessárias para acessar os inodes para todos os arquivos de um diretório. O sistema de arquivos antigo do Unix normalmente requer uma transferência de disco para buscar o inode para cada arquivo em um diretório, ganhando em casos de diretórios com poucos arquivos mas ficando muito atrás em casos de diretórios com dezenas de arquivos.

Outro recurso que precisa ser tratado com cuidado são os blocos de dados. Como os blocos de dados de um arquivo geralmente são acessados juntos, as política utilizadas tentam colocar todos os blocos de dados de um arquivo no mesmo grupo de cilindros, de preferência em posições que aproveitem a rotação do disco para economizar deslocamento do braço. O problema com a alocação de todos os blocos de dados no mesmo grupo de cilindros é que os arquivos grandes usarão rapidamente o espaço disponível no grupo de cilindros, forçando um derramamento em outros grupos. Além disso, o uso de todo o espaço em um grupo de cilindros faz com que alocações futuras de qualquer arquivo no grupo de cilindros também se espalhem para outras áreas. Idealmente, nenhum dos grupos de cilindros deve ficar completamente cheio. A solução heurística escolhida para ser implementada no UFS é redirecionar a alocação de blocos para um grupo de cilindros diferente quando o arquivo exceder 48 kilobytes e a cada megabyte depois disso. O grupo de cilindros recém-escolhido é selecionado dentre os grupos de cilindros que têm um número maior que a média de blocos livres restantes. Embora o grande depósito possa ser distribuído pelo disco, um megabyte de dados é normalmente acessível antes que uma longa busca seja realizada, e o custo de uma busca longa por megabyte é pequeno. Na época em que este sistema de arquivos foi desenvolvido, em 1984, um megabyte de dados era muita coisa. Entretanto, com a evolução da capacidade de armazenamento, que hoje ultrapassa os terabytes, esse valor pode parecer obsoleto.

III. Conclusão

O UFS, conforme foi descrito no trabalho atual, é um sistema de arquivos bastante antigo, com mais de 30 anos de idade. Sua proposta inicial foi evoluir o sistema de arquivos clássico do Unix, de forma a torná-lo configurável de acordo com o hardware em que se estava trabalhando.

O UFS trouxe algumas inovações não só de otimização como de estruturação para o antigo FS, inovações que eram bastante procuradas pelos usuários, mas como implementar elas obrigaria os sistemas antigos a restaurar seus sistemas de arquivos, não eram viáveis de serem aplicadas. Como o UFS já iria requerer essa restauração de qualquer jeito, algumas coisas bastante interessantes passaram a ser padrão nos sistemas Unix. Ainda foi tomado um cuidado adicional para que essas mudanças não quebrassem programas antigos que foram feitos para o FS anterior.

Essas inovações incluem coisas como nomes de arquivos longos, que a partir do UFS passaram a poder ser quase arbitrariamente longos, tendo na época de sua criação um tamanho máximo de 255 caracteres. Outra inovação são os links simbólicos, que já haviam sido concebidos de maneira similar anteriormente mas no UFS foram implementados literalmente como "um arquivo que contém um pathname". Além disso, o UFS criou a operação de renomear, para evitar que operações desnecessárias fossem usadas para isso.

Algumas distribuições de Unix adotaram o UFS como padrão, mas em geral esse sistema de arquivos foi adaptado de acordo com a necessidade de cada fornecedor, no fim gerando diversos outros sistemas de arquivos.

Referências

[1] Ken Thompson. Unix time-sharing system: Unix implementation. *The Bell System Technical Journal*, 57(6):1931–1946, 1978.

[2] Andrew S Tanenbaum and Herbert Bos. *Modern operating systems*. Pearson, 2015.

[3] Marshall K McKusick, William N Joy, Samuel J Leffler, and Robert S Fabry. A fast file system for unix. *ACM Transactions on Computer Systems (TOCS)*, 2(3):181–197, 1984.

[4] Make Tech Easier. Understanding the Difference Between a Disk, Drive, Volume, Partition, and Image, 2018.

[5] Wikipedia. Unix File System, 2019.