Um novo sistema de arquivos com journaling e transações atômicas

A quarta sinfonia de Hans Reiser

O novíssimo Reiser4 rapidamente se aproxima de sua *première* no kernel 2.6. Promete transações atômicas e velocidades de leitura maiores que seu predecessor, o ReiserFS 3, além de fazer melhor uso do disco rígido e ser extensível por meio de plugins.

POR MARCEL HILZINGER

Depois de um período de desenvolvimento de quase quatro anos, o sistema de arquivos Reiser4 chegou este ano à sua quarta versão, já disponível no site da Namesys [1]. A equipe de desenvolvimento de Hans Reiser reescreveu o Reiser4 do zero e está testando o desempenho e a estabilidade do herdeiro do ReiserFS. O processo inclui caça a bugs, que estão sendo massacrados. Entre outras coisas, o novo sistema de arquivos com journaling é capaz de transações atômicas e é extensível pelo uso de plugins. Neste artigo, faremos uma avaliação aplicando os patches necessários em um kernel da série 2.6 (veja também o quadro "Guia de Instalação"). O Reiser4 ainda não conseguiu uma cadeira na série oficial do kernel, mas já está em testes na árvore **mm**, de Andrew Morton.

Estrutura básica da tecnologia

De forma simples, o sistema de arquivos Reiser4 é composto de duas camadas: a camada de armazenamento e a camada semântica. Os objetos existem em ambas as camadas. O Reiser4 não distingue entre arquivos e diretórios, mas trata a ambos como objetos. Cada objeto no sistema de arquivos possui um *object ID* e uma chave, parcialmente gerada por referência ao object ID.

Como o nome sugere, as chaves abrem acesso ao sistema de arquivos. O Reiser4 usa chaves para posicionar objetos. É possível até usar uma chave para posicionar um setor específico de um arquivo. Na configuração básica, o Reiser4 usa chaves longas (por exemplo *10001:1:746f6d20776169:0:10278:0*) que englobam os elementos:

- Posição principal (Major Locality)
- Posição específica (Minor Locality)
- Ordem (Ordering)
- Não usado (sempre em 0)
- Identificador do objeto (Object ID)
- Deslocamento (Offset)

Enquanto a camada de armazenamento é responsável por guardar e organizar os dados, a camada semântica relaciona ("mapeia") os nomes de arquivo e as chaves entre si. O sistema de arquivos pode usar chaves para localizar os bytes corretos na camada

ReiserFS, mais que uma teoria

Hans Reiser trabalhou em sua teoria de dados e espaços de nomes por quase oito anos até que, influenciado pela teoria das "Estradas e Hidrovias", finalmente chegou à conclusão de que os sistemas de arquivos estão para a computação como as rodovias e vias fluviais estão para nossa civilização. Quanto mais e melhor conectado um sistema for, melhor é a habilidade de interagir e mais rápido pode-se avançar.

Reiser não é apenas um desenvolvedor que quer criar um sistema de arquivos, mas um teórico que considera um sistema de arquivos como a melhor comprovação de sua teoria. Como queria já de início criar seu sistema de arquivos para Linux, e como os recursos financeiros eram limitados, procurou por programadores bons, mas baratos. Assim, o desenvolvimento do ReiserFS iniciou-se em 1993 com Hans Reiser e uma equipe de jovens desenvolvedores russos.

O ReiserFS 3.6 é considerado estável desde o kernel 2.4.21. A Namesys, a companhia por trás do Reiser4 , colocou a nova versão sob intensivos testes destrutivos em 2003. O software foi liberado ao público em 2004 depois de removidos os bugs conhecidos.

de armazenamento. O *offset* informa a posição exata do byte dentro do objeto.

Plugins do sistema de arquivos

O Reiser4 usa plugins para desempenhar as operações com arquivos, incluindo seu próprio plugin interno para manipulação de arquivos e diretórios Unix. Cada arquivo e cada diretório possui um *plugin ID*, descrevendo uma coleção de métodos externos. Esses métodos podem ser usados para operações externas com os arquivos.

Operações externas são métodos de gerenciamento de arquivos que não fazem parte do sistema de arquivos. Permitem que programas externos interajam com partes inteiras do sistema de arquivos sem que seja necessário compilar um novo kernel. O Reiser4 possui mais de 30 plugins até agora. O comando *fsck.Reiser4 -l* mostra uma lista. Não haviam plugins externos disponíveis até o fechamento da edição. Um plugin de quota seria útil, já que o Reiser4 ainda não suporta quotas de disco.

Os plugins de arquivos e de "hash" são responsáveis pela organização dos objetos. Em contraste com a maioria dos sistemas de arquivos para Linux, o Reiser4 não reconhece os arquivos pela data de criação e sim de forma alfabética. No caso de arquivos com mais de quinze caracteres em seus nomes, os primeiros oito são significativos, seguidos pelo "hash" do nome completo.

O Reiser4 usa o plugin de hash para organizar diretórios. No momento, usa *cookies* construídos com o valor de hash e o contador de criação. Esta é uma solução temporária baseada no ReiserFS; no futuro, o plugin de arquivos ficará a cargo da classificação dos diretórios em ordem alfabética.

Módulos de segurança

Os plugins de segurança têm papel importantíssimo. Obviamente, isso tem a ver com o principal mecenas do Reiser4, a Agência de Projetos de Pesquisa Avançada para Defesa, o DARPA (Defense Advanced Research Projects Agency), vinculado ao Departamento de Defesa dos Estados Unidos. Um dos plugins cuida de arquivos extremamente grandes contendo muitas informações. Os sistemas de arquivos tradicionais aplicam regras de controle de acesso apenas a arquivos. É possível usar lis-

Guia de Instalação

Este guia abrange o kernel 2.6.5 e o *snapshot* oficial do Reiser4 datado de 26 de março de 2004 [2]. Versões mais novas do kernel possuem patches [3] que absolutamente não garantem o funcionamento correto se for usado um snapshot não correspondente! Use o snapshot apropriado para sua versão de kernel.

- Baixe os fontes do kernel a partir do servidor oficial [4] e descompacte-os em */usr/src/*.
- Copie o patch *all.diff.gz* a partir do site da Namesys no diretório do novo kernel */usr/src/linux-2.6.5/*.
- Vá para o novo diretório e aplique o patch com o comando *zcat all.diff.gz | patch -p1*.
- Há um erro na linha 570 do arquivo *fs/reiser4/as_ops.c*: uma declaração *->host* está faltando. Substitua a linha por:

```
mapping->host->dirtied_when= jiffies|1;
```

- Substitua *mapping->dirtied _when = jiffies|1;* na linha 477 do arquivo *fs/reiser4/plugin/tail.c* pela linha:

```
minode->dirtied_when = jiffies|1;
```

- Como root, execute o configurador do kernel com o comando *make xconfig* (os pacotes de desenvolvimento do QT 3 são necessários).
- Certifique-se de que a opção *Reiser4* foi selecionada em *File systems*. É possível compilar o suporte a Reiser4 como módulo ou incluído no kernel. Veja a figura 1.
- Depois de salvar a configuração, compile o kernel da maneira usual com o comando *make bzImage modules modules_install* e copie o novo kernel para o diretório apropriado (por exemplo, com o comando *cp arch/i386/boot/bzImage/boot/vmlinuz-reiser4*). Não sobrescreva o kernel original!
- Se sua distribuição usa um *RAM Disk* inicial (initrd) durante o boot (como o Suse, por exemplo), será preciso usar o comando *mkinitrd* para criar um novo RAM disk.
- Para acessar o novo kernel durante o boot, configure apropriadamente o gerenciador de boot em uso pelo sistema. No caso do Grub, por exemplo, deve-se editar o arquivo */boot/grub/menu.lst*.
- O arquivo */etc/fstab* não deve conter quaisquer opções específicas da distribuição que um kernel oficial não possa manipular. Por exemplo, o Suse 9.1 teve dificuldades com as opções *acl* e *user _xattr*. Em caso de dúvida, lembre-se de que tudo o que precisa são as opções *rw,exec*.
- Agora é possível reiniciar com o novo kernel. Após o boot, é necessário instalar os pacotes *libaal-0.5.0.tar.gz* e *reiser4progs-0.5.3.tar.gz*. O procedimento é o de sempre: *./configure && make && make install*. Não esqueça de rodar o comando *ldconfig* para carregar a biblioteca libaal antes de compilar os reiser4progs.
- Para criar partições Reiser4, use o comando */usr/local/sbin/mkfs.reiser4 Partition*. Depois disso, monte-as com o comando *mount*.

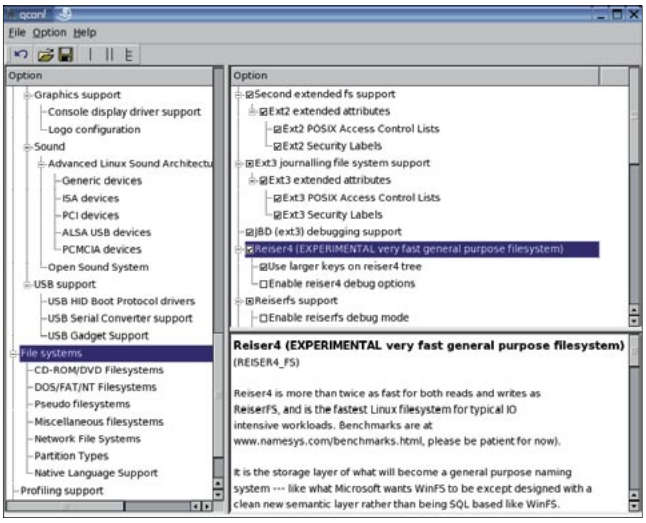

Figura 1: O kernel "remendado" oferece a opção de suporte a Reiser4 na seção *File systems*. A marca de OK significa que o suporte será compilado de forma monolítica no kernel. Um duplo-clique transformará a marca em um ponto: o Reiser4 funcionará como módulo.

Tabela 1: metadados importantes

Arquivo/diretório	Descrição	Explicação
bmap	Lista dos números de bloco associados a um objeto	Um número de bloco por linha
gid	ID do grupo primário de um objeto	Exemplo: 500
items	Lista de elementos que formam um objeto	Exemplo: *(2a:4:666f6f31000000:0:6d352:2) body length: 3*
key	Chave do objeto	Exemplo: *2a:4:666f6f31000000:0:6d352:0*
locality	Posição principal do objeto	A mesma para todos os objetos dentro de um diretório
new	Pseudo-arquivo para criação de objetos	
nlink	Número de links físicos a um objeto	
oid	Identificador (Object ID) do objeto	Exemplo: *97549*
pagecache	Estatística de cache do objeto	É possível ler o arquivo em bloco ou usar os sub-objetos *clean, dirty, io, locked* e *nrpages* para lê-los como diretório.
plugin	Plugins disponíveis para o objeto. Digite *mkfs.Reiser4 -l* para obter uma lista	
pseudo	Lista dos pseudo-arquivos em um objeto	
readdir	Lista de sub-objetos associados a um objeto. No caso de diretórios, lista seu conteúdo	
rwx	Controle de acesso para um objeto em forma numérica e texto simples	Exemplo: *0100755 -rwxr-xr-x*
size	Tamanho do objeto em bytes	Um novo diretório no Reiser4 ocupa apenas 2 bytes mais 1 byte para cada novo sub-objeto.
uid	ID do proprietário do objeto	Exemplo: *500*

tas de controle de acesso (access control lists – ACLs) para associar diferentes privilégios a diferentes usuários. Entretanto, se o usuário tiver acesso, os privilégios se aplicam ao arquivo todo.

O Reiser4 resolve esse problema dividindo o arquivo em um punhado de itens; por exemplo, o arquivo */etc/passwd* poderia ser dividido por referência a cada uma de suas linhas. Cada item da divisão teria seus próprios privilégios e poderia usar plugins diferentes. O Reiser4 também permite que delimitadores diferentes sejam aplicados aos itens dependendo do aplicativo acessando o arquivo. Aliás, os programas vêem cada item individual como um arquivo.

Arquivos e subdiretórios

Como mencionado antes, o Reiser4 não é uma extensão do ReiserFS mas um software novo, desenvolvido do nada. O projetista, Hans Reiser, insistiu em manter o código o mais simples possível. O quadro "ReiserFS – mais que uma teoria" dá detalhes sobre os primórdios da tecnologia.

O Reiser4 evita sobrecarregar o sistema de arquivos com atributos, como os atributos extendidos do Ext 2. A idéia é conseguir funcionalidade pela abordagem mais simples possível, usando arquivos. Para esse fim, cada objeto no Reiser4 possui um tipo de "sub-namespace" no diretório *metas*, que a chamada de sistema *readdir* não mostra. Entretanto, é possível entrar nesse diretório simplesmente digitando *cd metas*. Isso também vale para os arquivos: *cd nome_do_arquivo/metas* mudará para o diretório metas daquele arquivo. Entretanto, o arquivo deve ser executável para que isso funcione.

Os metadados (figura 2) no diretório *metas*0 são objetos anexados a outros objetos (os arquivos e diretórios reais). Eles não têm uma existência separada no disco (em contraste com os arquivos sob */proc* ou */sys*). O Reiser4 usa-os para implementar muitas das funções padrão do Unix, além de métodos especiais de segurança e compressão.

Por exemplo, você pode alterar o dono de um arquivo sem precisar do comando *chown*, ou usar um simples *cat* para ver os privilégios de um diretório (figura 2, embaixo). A tabela 1 descreve rapidamente os metadados mais importantes no diretório *metas*.

Um sistema de arquivos vivo!

Sistemas de arquivos modernos para Unix, como o ReiserFS, o XFS e o JFS, são baseados em estruturas de árvore. Precisamos distinguir entre as estruturas B+ e B- para entender a teoria a partir deste ponto. Enquanto estruturas B+ apenas armazenam ponteiros e informações administrativas em seus nós internos, as árvores B- usam os nós para o armazenamento de dados. Como B+ armazena dados apenas nas "folhas" das árvores, uma estrutura B+ precisa de muito mais folhas para armazenar a mesma quantidade de dados que uma árvore B-. Posto isso, a estrutura B+ gerencia os nós internos de maneira mais eficiente, pois contém apenas ponteiros para os dados.

O ReiserFS e o Reiser4 usam apenas árvores do tipo B+. A maior diferença entre as árvores balanceadas tradicionais, como as usadas no ReiserFS 3.6, e as árvores "balançantes" usadas no Reiser4 é a maneira como os nós internos e os objetos binários muito grandes (os famigerados BLOBs – Binary Large Objects) são tratados.

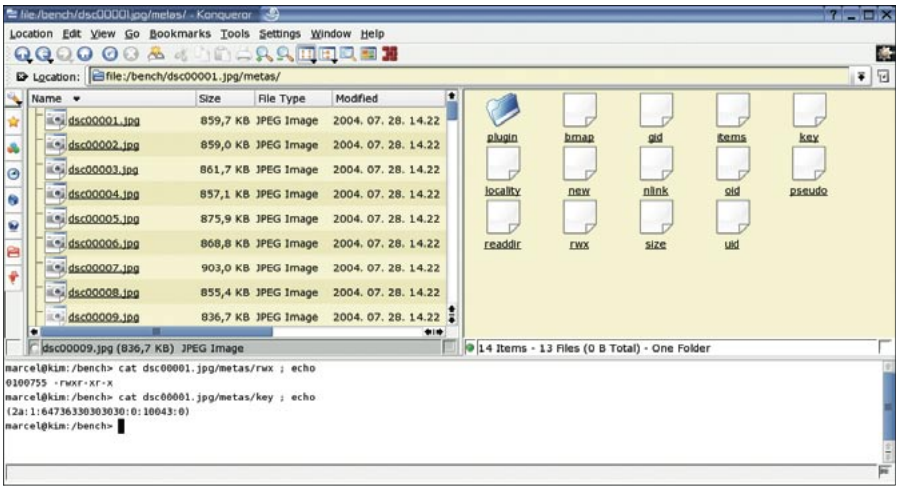

Figura 2: Um diretório com imagens numa partição Reiser4 (à esquerda na janela). Não é possível chegar aos metadados com o mouse. É necessário adicionar *metas* à barra de endereços para revelar os pseudo-arquivos da imagem *dsc00001.jpg*. O diretório de plugins também é mostrado à direita.

BLOBs são arquivos que precisam de mais espaço do que um nó final (ou seja, uma "folha" da árvore) pode oferecer. Para armazenar BLOBs, os bancos de dados relacionais e o ReiserFS usam uma folha em um nó para armazenar ponteiros para outras folhas. Isso cria nós internos adicionais e as chamadas "árvores altamente balanceadas" (nas quais o caminho a partir de qualquer folha em direção à raiz deveria ter o mesmo comprimento) ficam desbalanceadas. Veja a figura 3.

O Reiser4 armazena os BLOBs diretamente nas folhas, tratando-os como qualquer outro tipo de dado(figura 3, embaixo). Isso minimiza o número de nós internos e mantém a árvore balanceada. Seu site afirma que o código fonte do kernel 2.4.18 ocupa 1629 nós internos no ReiserFS, em comparação com meros 164 no Reiser4 .

Em um computador comum com Reiser4 , todos os nós internos da árvore do sistema de arquivos devem caber na memória. Se não for possível evitar o uso da memória virtual em disco (swap), o Reiser4 primeiro armazena os nós mais distantes da raiz e utiliza mais nós no espaço livre à direita.

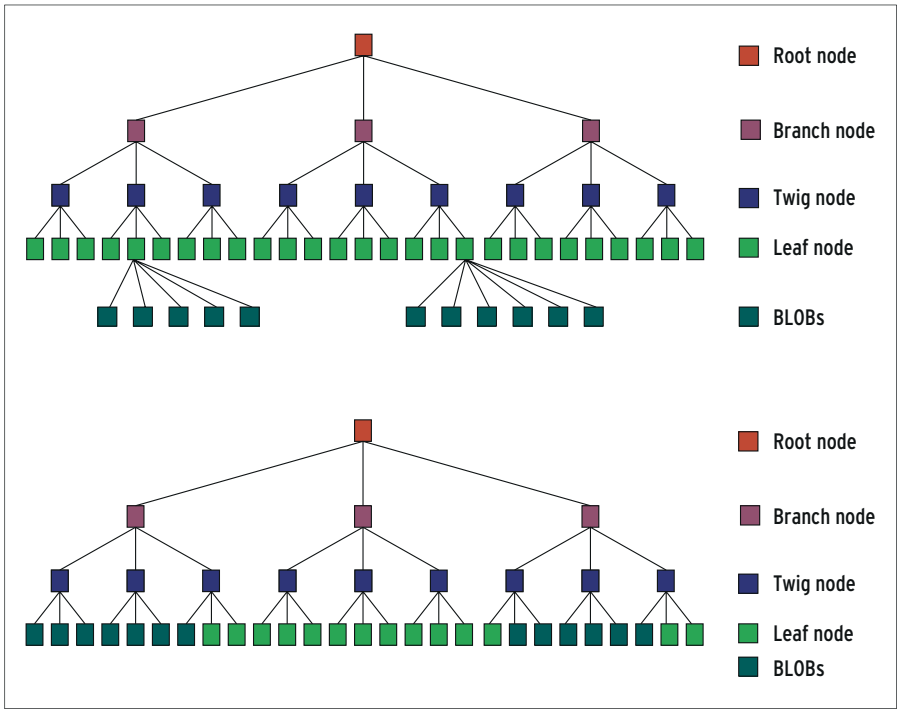

Figura 3: Uma árvore balanceada tradicional (em cima), da forma como é usada pelo ReiserFS 3.6. Aqui, a maneira de armazenar BLOBs usa uma folha que assume o papel de nó interno. Contrastando, uma árvore Reiser4 (embaixo) guarda os BLOBs no nó da própria folha, ajudando a manter mínimo o número de nós internos.

Árvores dançarinas

O balanceamento contínuo da árvore adicionaria complexidade desnecessária caso todos os dados estivessem na memória principal, mas faz todo o sentido do mundo liberar espaço em disco e, dessa forma, evitar operações de entrada/saída (I/O). Por isso, o Reiser4 comprime os dados armazenados na memória imediatamente antes de serem gravados no disco. Essa abordagem melhora o desempenho de forma significativa. As árvores balanceadas tradicionais, como as usadas no ReiserFS, são tipicamente mais compactas mas tendem a precisar, com muita freqüência, de transporte de dados dentro da árvore. A perda de compactação é uma das desvantagens das "árvores dançantes" usadas no Reiser4. Um reempacotador planejado para a versão 4.1 ajudará a suavizar o problema.

O ReiserFS atribui novos nós ao criar um número de bloco, enquanto o Reiser4 espera até que o sistema grave os dados da memória no disco. Essa característica, "chupada" do XFS, implica em que arquivos apagados antes de sua gravação no disco causam impacto zero no sistema de arquivos, resultando numa melhora brutal no desempenho geral.

Transações atômicas

A habilidade de executar transações atômicas é um requisito tradicional num sistema de arquivos. Por exemplo, imaginemos que uma falha no fornecimento de energia tenha ocorrido durante o procedimento de movimentação (ou seja, cópia seguida de apagamento) de um arquivo qualquer, do diretório A para o diretório B. Em um sistema de arquivos atômico, o arquivo estará tanto no diretório A quanto no diretório B quando a força voltar e sob nenhuma circunstância estará parcialmente em A ou B. A prática atual, seguida pelos sistemas de arquivos populares, é usar o utilitário *fsck* para verificar o sistema de arquivos depois da falha e salvar no diretório *lost+found* quaisquer fragmentos de dados encontrados.

O Reiser4 usa essa abordagem atômica. Apesar de ser possível que existam fragmentos de arquivos caso uma falha ocorra durante uma cópia, os fragmentos conterão dados do arquivo copiado, apenas, em vez de lixo binário. Isso é importante por razões de segurança, pois arquivos corrompidos podem conter informações sigilosas.

Dependendo das circunstâncias, o Reiser4 gravará os arquivos uma vez (da origem para o destino) ou duas (da origem para o *journal* e dali para o destino). Para proporcionar operações atômicas em transações simples, o Reiser4 simplesmente pula a parte do journal, gravando direto no destino.

Se o Reiser4 precisar copiar um arquivo do diretório A para o B, primeiro registrará a operação no LOG, que é a área de armazenamento do journal, e então indicará os nós que apontam para B no LOG a partir dali. O journal recém-criado vagueia até uma nova posição, o que explica seu nome – "wandering log", ou "registro andarilho".

Durante a transação, o Reiser4 mantém os blocos modificados onde estão até que o sistema confirme que a transação foi completada. Depois de receber essa mensagem, o Reiser4 complementa a transação. O número de operações de escrita depende das mudanças efetuadas.

No caso de pequenas mudanças em grandes arquivos, é bom usar duas operações de escrita. Isso significa que o arquivo fica intocado enquanto as alte-

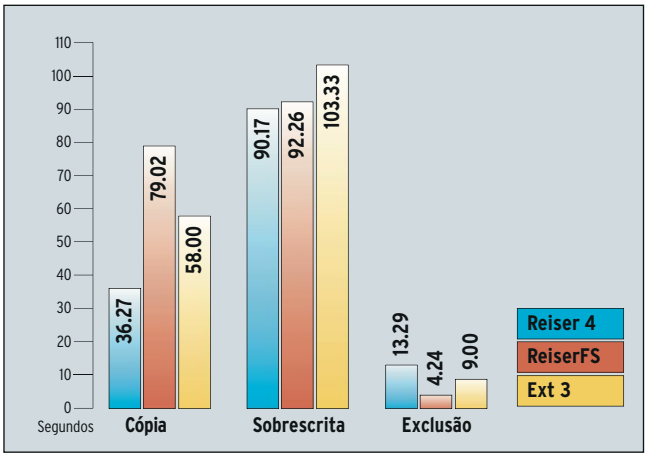

Figura 4: A primeira operação de gravação dos fontes do kernel mostra que o Reiser4 é quase duas vezes mais rápido que o Ext3. Os candidatos estão mais ou menos empatados em operações de escrita em arquivos já existentes (overwrite). O ReiserFS apaga arquivos mais rapidamente que qualquer um dos outros.

rações são gravadas. No caso de mudanças maiores, vale mais a pena alterar o nó e deixar o resto do trabalho para o reempacotador. Esta operação ainda não está presente, pois o reempacotador ainda não foi implementado. Portanto, o Reiser4 sempre fará duas operações de gravação.

Desempenho

Como é um sistema de arquivos de propósito geral, colocamos o Reiser4 em teste com algumas ferramentas de medição de desempenho para PCs comuns. Nossa máquina de testes possuía 256 Mbytes de RAM e um disco rígido de 40 GBytes, um Maxtor Baracuda ATA133. Usamos um SuSE Linux 9.1 com kernel 2.6.7-mm4 e aplicamos um patch recente do Reiser4 , datado de 6 de julho de 2004 [3].

Usamos o *fs_bench* de [5] como benchmark do sistema de arquivos. O *fs_bench* combina as técnicas Bonnie++ [6] e IOzone [7] de medição de desempenho. Além desse teste, também usamos os resultados da ferramenta *slow.c*, disponível no próprio site da Namesys, e nossos próprios testes.

O Reiser4 entra para a história com resultados excelentes, em particular quando manipula arquivos pequenos (figura 4). O novo sistema de arquivos não é apenas rápido. Também é avarento: o kernel 2.6.7 ocupou apenas 371 MBytes de disco no Reiser4. No ReiserFS, foram necessários 431 MBytes e no Ext3 o espaço ocupado foi de 446 Mbytes.

Um detalhe que percebemos foi que o disco rígido emite muito menos ruído durante os testes com Reiser4 em comparação com os outros sistemas de arquivos. Os desenvolvedores da Namesys realmente souberam aproveitar os resultados de seus estudos sobre o movimento das cabeças de gravação em discos rígidos, otimizando o Reiser4 para operações de gravação.

Reiser4, o copião turbinado

A figura 4 deixa claro que o Reiser4 vence "com um pé nas costas" o teste de cópia de código-fonte. Os três candidatos chegaram a um empate técnico no teste de sobrescrita (overwrite), embora o Reiser4 seja ligeiramente mais rápido que seus competidores.

Entretanto, o ReiserFS apaga arquivos mais rápido do que qualquer um dos outros candidatos. O Reiser4 ainda tem chão a percorrer até alcançar esse nível. Isso pode ser atribuído à estrutura interna do sistema de arquivos e parece causar o efeito inverso se os arquivos forem da ordem de gigabytes: o Reiser4 levou apenas 1,5 segundos para apagar um arquivo de 6 Gbyte. O ReiserFS levou 12 segundos e o Ext3 intermináveis 14 segundos.

Os testes Bonnie++ e IOzone resultaram em dados curiosos. O Reiser4 é extremamente rápido em muitos aspectos, mas há alguns detalhes em que perde para o velho ReiserFS. Por exemplo, o apagamento de arquivos nos testes de criação seqüencial e criação aleatória com o Bonnie++ mostraram vantagens para o ReiserFS. O Reiser4 continua a sobrecarregar brutalmente a CPU: muitos testes contra o Ext3 e mesmo contra o ReiserFS foram mascarados por isso.

O tempo total para término dos testes é um bom parâmetro para estimar o quanto um sistema de arquivos é bem (ou mal) balanceado. Os resultados nas figuras 5a e 5b indicam que, baseado apenas no tempo total, o Reiser4 é o campeão incontestável.

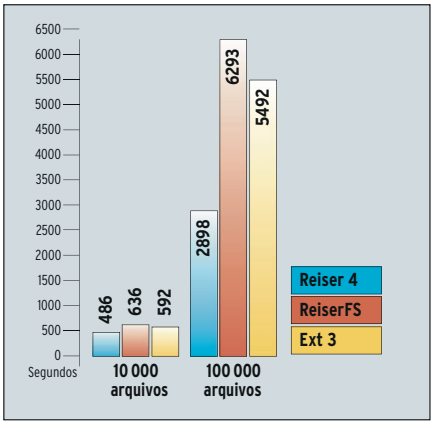

Figura 5a: Quanto mais arquivos de 1 GByte são manipulados, melhor o desempenho do Reiser4 no teste Bonnie++, em comparação com os outros sistemas de arquivos.

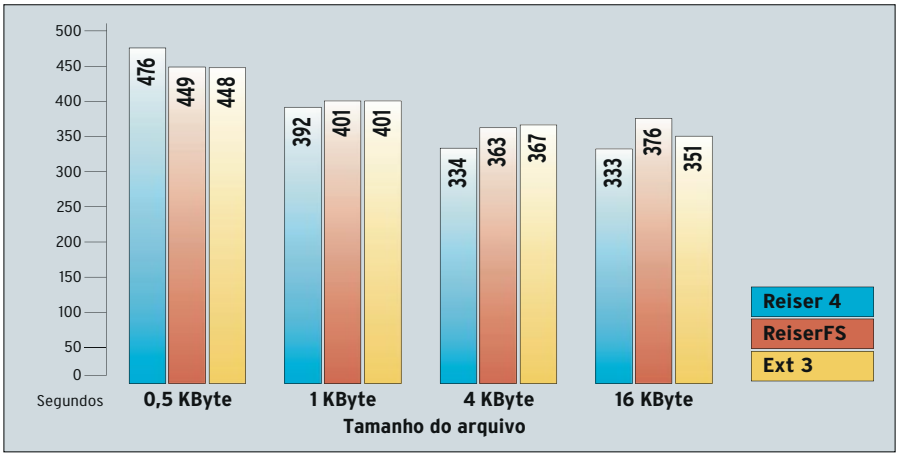

Figura 5b: O IOzone usa um arquivo de 1 GByte para medir o desempenho. O Reiser4 é mais lento que o ReiserFS ou o Ext3 com arquivos minúsculos, mas tamanhos de 4 ou 16 KBytes são mais relevantes para aplicações práticas.

Rapidez e seriedade

O teste *slow.c* basicamente demonstra que o Reiser4 é mais rápido que o ReiserFS 3 quando deve gravar múltiplas *streams* de dados ao mesmo tempo (figura 6). Em comparação com a leitura ou gravação de um único arquivo, operações com múltiplas streams são um verdadeiro soco no estômago do ReiserFS 3. Em contrapartida, o desempenho do Reiser4 mantém-se mais ou menos inalterado.

O Ext3 consegue manter-se cabeça-a-cabeça com o Reiser4 em operações de gravação, mas fica muito para trás em operações com quatro streams paralelas de dados. A taxa de transferência real com o Reiser4 fica surpreendentemente próxima à máxima taxa física possível do disco rígido, no caso 27.6 MBytes (de acordo com o *hdparm -t*). No futuro, espera-se que o novo sistema de arquivos quebre recordes de velocidade com o uso de plugins de compressão e atinja taxas de transferência maiores que a velocidade física do disco.

Não há maneiras confiáveis de se medir o desempenho para montar uma partição ou para criá-la; e isso pode ser o motivo principal de se escolher um sistema de arquivos específico para máquinas de produção. O Reiser4 ainda tem chão pela frente nesse quesito, levando cerca de 12 segundos para montar um volume de 200 Gbytes. Em comparação, o ReiserFS leva 4 segundos e o Ext3 menos de 1 segundo, para a mesma partição de 200 GBytes. Em um resultado um pouco melhor, o Reiser4 levou apenas 1 segundo para criar uma partição, enquanto o ReiserFS precisou de quatro segundos.

Essa diferença pode ser atribuída ao fato de que o Reiser4 cria apenas os nós necessários ao sistema de arquivos para marcar sua existência. Assim, o sistema de arquivos ocupa apenas um espaço mínimo no sistema. Em contraste, o ReiserFS cria uma área de arquivos LOG e ocupa aproximadamente 30 MBytes na recém-criada partição. Criar uma partição de 200 GBytes em Ext3 lembrou-nos dos velhos (e ruins) tempos do DOS. O Ext3 levou nada menos que 5 **minutos** para criar as tabelas de inodes Ext2 e arquivos LOG.

Finalmente, expusemos o sistema de arquivos Reiser4 a vários testes destrutivos. Pressionamos o botão de reset do computador para reiniciar "na marra" a máquina enquanto o Reiser4 estava ocupado com várias operações. Embora tenhamos repetido o teste umas 15 vezes, o Reiser4 não mostrou sinais de corrupção, todos os arquivos testados estavam legíveis; o sistema de arquivos não sofreu dano algum.

O Reiser4 possui um bom punhado de novas características e seus testes de desempenho parecem bem promissores. A única pergunta que fica é se o Reiser4 conseguirá resistir às expectativas da comunidade Linux. Por quase dois anos a Namesys anunciou em seu site que o Reiser4 seria lançado "ainda neste verão". Agora, depois de passar pelos testes internos, o sistema de arquivos precisa passar pelo crivo dos usuários da comunidade, que farão seus próprios testes. A Linspire anunciou que integrará o Reiser4 em sua distribuição Linux assim que possível e a SuSE também mostrou interesse em incluir suporte a Reiser4 tão logo alcance um nível aceitável de estabilidade.

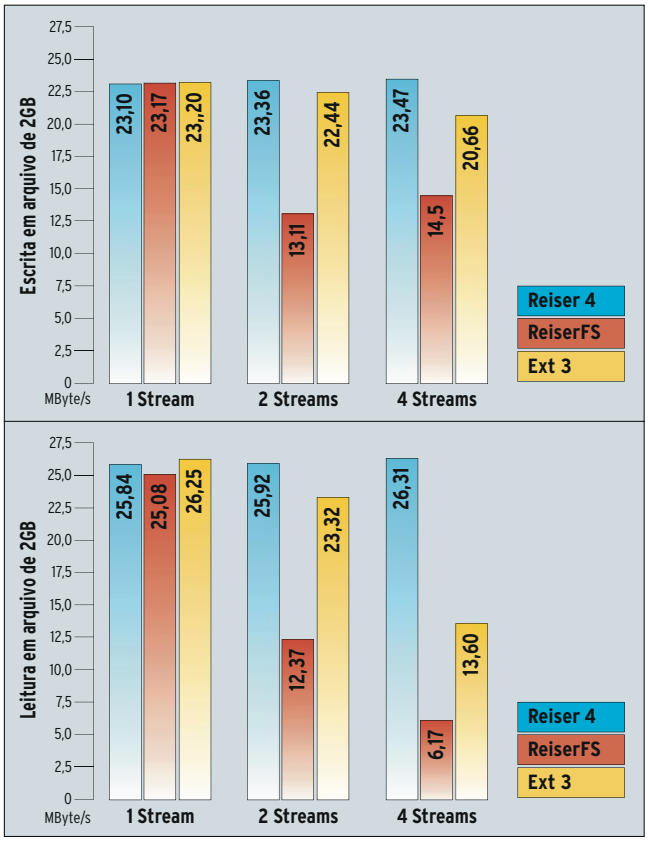

Figura 6: O teste de desempenho *slow.c* testa a velocidade do sistema de arquivos em operações de escrita em paralelo de múltiplas streams simultâneas de dados. Há traços desprezíveis de impacto no desempenho do Reiser4, enquanto o Ext3 mostra-se mais vulnerável e o ReiserFS é notadamente mais lento.

Pacotão de Natal

No momento a Namesys está procurando por patrocinadores, pois o dinheiro da DARPA só pode ser usado no desenvolvimento de itens relacionados à segurança. Portanto, se você pretende instalar o Reiser4 em seu computador neste Natal, considere com carinho a possibilidade de fazer uma doação para a Namesys. ■

INFORMAÇÕES

[1] Reiser4: *http://www.namesys.com*

[2] Kernel do Linux versão 2.6.5: *http://www.namesys.com/snapshots/LATEST/*

[3] Patches para o kernel atual: *http://www.namesys.com/auto-snapshots/*

[4] Kernel do Linux: *http://www.kernel.org*

[5] Fsbench: *http://fsbench.netnation.com*

[6] Bonnie++:*http://www.coker.com.au/bonnie++/*

[7] IOzone: *http://www.iozone.org*

[8] Conceituação e descrição da tecnologia Reiser4: *http://www.namesys.com/v4/reiser4_the_atomic_fs.html*

SOBRE O AUTOR

Marcel Hilzinger estudou história na universidade. Trabalha no escritório da SuSE GmBH em Budapeste desde 2001. Entre outras coisas, traduziu toda a documentação do SuSE Linux para o húngaro.