Validação experimental de sistemas de arquivos baseados em *journaling* para o sistema operacional Linux

Leonardo Garcia de Mello

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – CEP 91.501-970 – Porto Alegre – RS – Brazil
lmello@inf.ufrgs.br

Abstract. *Operating systems are important to high availability, and those that have secure filesystems relatively independent of human actions for recovery should be preferred. One of the approaches to gain high availability for filesystems are those based upon journaling, or meta-data logging. There is a series of journaling filesystems for Linux operating system, as ext3, JFS, ReiserFS and XFS. This works aims to propose a way to experimentally validate journaling filesystems efficiency. To do so, the techniques of Software Fault Injection are used for validation and the system target is an XFS implementation..*

Resumo. *Sistemas operacionais são importantes para alta disponibilidade, sendo preferível o uso daqueles que possuam sistemas de arquivos seguros e relativamente independentes de ações por agentes humanos para a recuperação. Uma das abordagens para obter-se uma alta disponibilidade em sistemas de arquivos é a do tipo journaling, ou log de metadados. Existem vários sistemas de arquivos para o Linux baseando-se nela, tais como ext3, JFS, ReiserFS e XFS. Este artigo propõe uma metodologia de validação experimental para avaliar a eficiência do mecanismo de journaling. As técnicas de Injeção de Falhas por Sofware são utilizadas para validação e o sistema alvo é uma implementação do XFS.*

1. Introdução

O termo comercialmente conhecido como "alta disponibilidade" (também referenciado como *HA*, de *High Availability*) representa uma característica de sistemas computacionais projetados para evitar ao máximo as interrupções, planejadas ou não, na prestação de serviços. Em alta disponibilidade, o ideal é haver poucas falhas e, mesmo quando estas acontecerem, que o seu tempo médio de reparo (ou *MTTR*, de *Mean Time To Repair*) seja tão pequeno quanto possível.

Atualmente o Linux é muito utilizado em servidores [BAR 2000], em um cenário onde os requisitos quanto à disponibilidade geralmente são bastante elevados. Praticamente todas as plataformas atuais dispõem de sistemas de arquivos com alta disponibilidade [SEL 2000], mas o ext2 (sistema de arquivos convencional para o Linux) não possui esta característica. Para uso em aplicações críticas, existe uma série de sistemas de arquivos baseados em *journaling* para o Linux, tais como ext3, JFS, ReiserFS e XFS.

Este trabalho propõe uma metodologia de validação experimental para avaliar a eficiência do mecanismo de *journaling* para sistemas de arquivos. Para isso, a técnica de validação empregada é a da injeção de falhas por *software* [HSU 97], e o sistema alvo é uma implementação do XFS.

No grupo de Tolerância a Falhas da UFRGS realiza-se a validação experimental de mecanismos que permitam maior tolerância a falhas para sistemas operacionais. Desta maneira, este artigo apresenta um trabalho sobre injeção de falhas em sistemas de arquivos baseados em *journaling* para o Linux. É apresentado o modelo de falhas sob o qual foi feito o desenvolvimento de uma ferramenta para injeção de falhas em sistemas de arquivos baseados em *journaling* – o FIJI.

2. Integridade para dados e metadados

No Unix, informações são organizadas em disco na forma de dados e metadados. Durante as operações envolvendo dados e metadados, é preciso que a representação do sistema de arquivos nos discos seja mantida consistente – isso mesmo após a ocorrência de falhas. Em sistemas de arquivos como o ext2, construídos com alocação baseada em blocos, o sistema de arquivos pode ficar inconsistente após a ocorrência de falhas porque dados e metadados são gravados em disco de forma

assíncrona por razões de desempenho. Sempre que algo assim acontece, é preciso executar algum utilitário de verificação - como o fsck [BAR 2000].

O fsck realiza uma série de passos ao longo de todo o sistema de arquivos de um dispositivo, para validar as suas entradas e assegurar-se de que blocos alocados em disco estão todos sendo referenciados corretamente. Mas infelizmente, para discos de tamanho muito grande, a execução do fsck pode consumir um tempo bastante elevado. Em uma máquina com muitos *gigabytes* em arquivos, por exemplo, a execução do fsck pode demorar até 10 horas ou mais [BAR 2000]. Além disso, conforme a severidade do caso, pode ser necessária a presença física do administrador do sistema para informar a senha do usuário *root* e executar manualmente o *fsck.*

Assim, levando-se em conta essas dificuldades, surgiram as propostas de sistemas de arquivos para alta disponibilidade. Por meio deles, é possível diminuir a chance de serem introduzidas inconsistências nos sistemas de arquivos e reduzir o tempo médio de reparo (ou *MTTR*) na ocorrência de falhas. Atualmente as principais técnicas empregadas para auxiliar a obter alta disponibilidade em sistemas de arquivos Unix são o *journaling* e o *Soft Updates* [SEL 2000].

Enquanto a técnica de *Soft Updates* apenas é utilizada em sistemas BSD, a técnica de *journaling* baseia-se na redundância para aumentar a confiabilidade dos dados e metadados - mas sem aumentar significativamente os custos de hardware. Ela já é adotada por sistemas de arquivos em sistemas operacionais para plataformas diversas, tais como Solaris, AIX, Digital UNIX, HP-Ux, Irix e Windows NT [SEL 2000]. Mas apesar de os sistemas de arquivos baseados em *journaling* haverem sido adotados quase como um padrão pela indústria de software, atualmente não se encontram publicações sobre avaliação de medidas da sua disponibilidade.

3. Sistemas de arquivos baseados em journaling

Sistemas de arquivos baseados em *journaling* fazem um controle sobre as mudanças realizadas ou nos metadados (para JFS, ReiserFS e XFS), ou nos dados e metadados (para ext3) associados a um sistema de arquivos. A idéia consiste em tratar diferentemente dados e metadados, usando uma área dedicada em disco (o *log*, ou *journal*) para manter um histórico das mudanças.

Estes sistemas de arquivos implementam uma política de *write-ahead logging*, fazendo com que registros sejam armazenados no *log* antes de as operações serem efetuadas [SEL 2000]. Este tipo de sistema de arquivos pode ser empregado com desempenho ótimo para aplicações que lidem com arquivos pequenos e façam uso freqüente da chamada de sistema *sync*. Isso acontece porque várias alterações nos metadados são transferidas para o disco através de uma única operação, que acrescenta várias transações em uma mesma área [SEL 2000].

Uma grande vantagem da abordagem baseada em *journaling* é de que ela facilita obter alta disponibilidade em sistemas de arquivos já existentes. Exemplo disso é o resultado que pôde ser obtido com o sistema de arquivos ext2, que pôde ser reimplementado como sendo o ext3 [TWE 2003].

Quanto aos tempo de recuperação, para sistemas de arquivos baseados em *journaling* eles são bastante reduzidos. A tarefa de verificação consiste apenas em inspecionar as transações pendentes do *log* - ao invés de percorrer todos os blocos buscando inconsistências. Com isso, na ocorrência de defeitos, o sistema de arquivos pode ser levado a um estado consistente pela aplicação das transações pendentes no *log* - ao invés de ser necessário inspecionar toda uma unidade de disco com o *fsck*.

3.1. O sistema de arquivos XFS

O XFS possui uma grande confiabilidade, pois foi criado ainda em 1994 pela SGI para substituir o EFS [XFS 2003]. Ele é o mais antigo entre os sistemas de arquivos conhecidos, e a ênfase do seu projeto foi na capacidade de trabalhar com arquivos bastante grandes - da ordem de "*terabytes*".

O XFS pode trabalhar usando um tamanho de bloco variando entre 512 bytes e 64 kbytes, provendo o suporte para sistemas de arquivos distribuídos (incluindo NFS versão 3), ACLs no padrão POSIX 1003.e, e quotas para usuários e grupos.

O estado de um sistema de arquivos XFS é o resultado da combinação de informações localizadas em três lugares diferentes: disco, memória e *log* do *journaling*. O sistema de arquivos XFS somente estará em um estado consistente após ele ter sofrido *shutdown*, onde todos os dados residentes em memória (*buffers* e *cache*) são gravados em disco e todas as entradas do *log* de transações são aplicadas no sistema de arquivos.

4. Injeção de falhas no XFS

Para a realização de um experimento de injeção de falhas, faz-se necessária a definição de um modelo de falhas. Um modelo de falhas permite-nos repetir um experimento tantas vezes quantas forem necessárias para atingir o objetivo. Para este caso, o objetivo é detectar erros que possam ocorrer durante a utilização de um sistema de arquivos baseado em *journaling*.

A maioria dos mecanismos de tolerância a falhas disponíveis nos sistemas de arquivos baseados em *journaling* consideram principalmente as falhas de hardware transientes (mais especificamente de falta de energia), onde assume-se um modelo de *crash* para o sistema. Isto representa uma situação em que o sistema de arquivos estaria sendo utilizado e ocorreria a falha, fazendo com que a máquina fosse reiniciada sem que ocorresse um *flush* das informações em memória (tais como *buffers* e *cache*) ou das transações no *log* (elementos que, conforme a Seção 3.1, definem o estado do sistema de arquivos).

Este modelo de falhas define o conjunto de falhas que deve ser gerado pela ferramenta FIJI.

5. FIJI

A ferramenta FIJI (***F**ault **I**Njector for **J**ournaling f**I**lesystems*), desenvolvida no PPGC da UFRGS, é baseada nos recursos de depuração do sistema operacional Linux (os quais permitem a interceptação e manipulação de chamadas de sistema) para a implementação de um injetor de falhas específico para sistemas de arquivos baseados em *journaling*. Manipular os parâmetros de chamadas de sistema (ou *system calls*) equivale a alterar as requisições feitas ao sistema operacional.

O FIJI é implementado como um processo concorrente à aplicação alvo, usando os recursos oferecidos pelo sistema operacional Linux através da chamada de sistema *ptrace*. A chamada de sistema *ptrace* permite executar um processo de três maneiras: passo-a-passo, usando *breakpoints* ou executá-lo até a próxima chamada de sistema.

Conceitualmente, o FIJI localiza-se entre a aplicação alvo e o sistema operacional. As falhas injetadas pelo FIJI estão de acordo com o modelo de falhas descrito anteriormente, e encontram-se especificadas no código-fonte da própria ferramenta.

Mais especificamente, o que procura-se fazer através do injetor de falhas FIJI é manipular uma quantidade de informações diferente daquela que foi solicitada originalmente. Para isso, é feita uma alteração nas chamadas de sistema *write* para mudar a quantidade de bytes que deve ser gravada (registrador EDX). Um parâmetro deve ser informado para o FIJI pela linha de comando, informando dentro de quantos segundos as chamadas de sistemas *write* deverão começar a ser suprimidas.

6. Arquitetura para injeção de falhas

Por meio do injetor de falhas FIJI, espera-se poder caracterizar um ambiente de falhas como o que está definido na Figura 1. De acordo com esta figura, o sistema alvo é uma máquina com uma partição Linux. Neste sistema alvo, foi criada uma partição para os testes. Esta partição deverá ser formatada com o utilitário mkfs.xfs [XFS 2003].

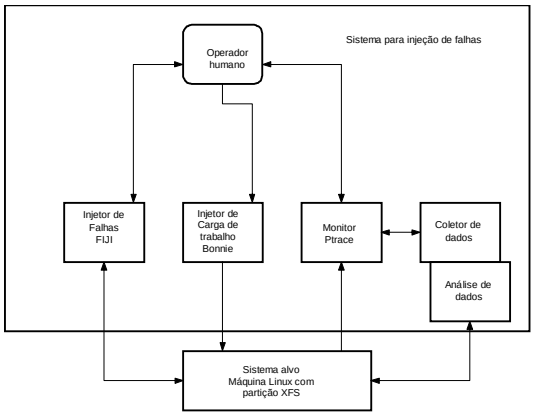

Figura 1 – Ambiente ideal para injeção de falhas

O componente controle é um ser humano, quem coordena a realização dos experimentos. A função de injetor de falhas é executada pela ferramenta FIJI. Desta maneira, é possível simular a ocorrência de falhas do tipo *crash*.

A função de gerador de carga de trabalho é feita pelo utilitário *bonnie*, que deve gerar um *workload* correspondente às aplicações reais - tais como arquivos muito grandes (mais de 130 Mb), arquivos pequenos (menos de 600 Kb), servidor de arquivos, servidor de email, etc.

Espera-se poder determinar a cobertura de falhas e o tempo de recuperação para o sistema de arquivos XFS.

7. Conclusões

Os trabalhos em andamento pelo grupo envolvem os sistemas de arquivos baseados em *journaling* para uma validação de suas propriedades relacionadas com alta disponibilidade. Atualmente estão sendo conduzidos experimentos para consolidar o ambiente de acordo com uma metodologia desenvolvida pelo grupo de tolerância a falhas da UFRGS e aplicada com sucesso em sistemas de banco de dados.

Os experimentos estão direcionados para XFS em Linux, mas espera-se que a metodologia seja suficientemente genérica e portável para permitir a validação de uma vasta gama de sistemas de arquivos baseados em *journaling*. Dados obtidos preliminarmente com os experimentos manuais comprovam a redução do tempo de recuperação em pelo menos uma ordem de grandeza no tempo no sistema XFS se comparado com sistemas de arquivos convencionais, como o ext2, sob uma mesma carga de trabalho.

8. Referências bibliográficas

[BAR 2000] BAR, Moshe. **Linux Kernel Internals**. New York: McGraw-Hill, 2000, 351 p.

[HSU 97] HSUEH, Mei-Chen; TSAI, Timothy K.; IYER, Ravishankar K. Fault-Injection Techniques and Tools. **Computer**, v. 30, n. 4, Apr 1997, p.75-82.

[SEL 2000] SELTZER, M. I.; et. al. *Journaling* versus soft-updates: Asynchronous meta-data protection in file systems. In: USENIX ANNUAL TECHNICAL CONFERENCE, 2000. **Proceedings...** San Diego, California, USA, 2000. Disponível em: < http:// www.lcs.ece.cmu.edu/~soule/papers/seltzer.pdf>. Acesso em: ago. 2003.

[TWE 2003] TWEEDIE, Stephen C. Ext3, *Journaling* Filesystem for Linux. In: OTTAWA LINUX SYMPOSIUM, 2000. **Proceedings...** Disponível em: <http://olstrans.sourceforge.net/release/OLS2000-ext3/OLS2000-ext3.html>. Acesso em: out. 2003

[XFS 2003] Sistema de arquivos XFS para Linux. Disponível em: <http://xfs.sourceforge.net>. Acesso em: ago. 2003.