Domain Disk Free, Uma Ferramenta Para Gerência De Disco Em Ambientes Distribuídos

Marco Aurélio Medina de Oliveira *Vitor Hugo Furtado*
Carlos Fernando Bella Cruz *Paulo Lício de Geus*

Instituto de Computação
Universidade Estadual de Campinas
13081-970 Campinas, SP
E-mail: {marcoamo, furtado, cfbcruz, paulo}@dcc.unicamp.br

Resumo

A fim de facilitar a administração das redes de computadores Unix, o administrador necessita de uma série de ferramentas que o auxilie em seu trabalho. No caso dos discos, é muito importante para o administrador a obtenção destas informações de maneira simples e rápida de todas as máquinas da rede. Com esse objetivo foi desenvolvido o utilitário ddf. Neste artigo descrevemos sua arquitetura e funcionamento, e relatamos alguns resultados.

Abstract

To make the administration of Unix computer networks easier, the manager needs a set of tools to help with the work. One very important aspect to monitor on these networks is information about all disks on the net, to be obtained preferably in a fast and simple way. That was the motivation to develop the ddf tool. In this paper, we describe its architecture, operation, and report some results.

1 Introdução

Atualmente, as redes de computadores vêm crescendo cada vez mais na quantidade e na qualidade de seus serviços. Com isto, há também um acréscimo na complexidade de sua administração. Para suprir esta necessidade, tem surgido uma série de ferramentas que visam facilitar a administração dessas redes.

Uma das dificuldades existentes é a gerência do espaço em disco em redes de maior porte. Freqüentemente, é necessário ao administrador informações a respeito dos discos. Por exemplo, quando for efetuada a instalação de algum software, ou a criação de um novo grupo, é necessário o conhecimento do espaço livre dos discos disponíveis na rede para saber se é possível a ocupação do espaço que a tarefa venha a requerer, e em qual disco alocar o espaço. Para obter estas informações no sistema Unix, existe o utilitário *df* (*disk free*) que costuma ser utilizado satisfatoriamente.

Entretanto, quando se tem um ambiente distribuído com muitas máquinas com disco, o utilitário df deixa de ser adequado, pois ele fornece informações apenas a respeito do disco local à máquina consultada. Para que o administrador tenha informações sobre todos os discos existentes no sistema, ele é obrigado a executar o comando df em todas as máquinas e reunir todas essas informações para escolher o disco onde, por exemplo, será criado um novo grupo ou instalado um novo *software*.

Percebe-se claramente a necessidade de obter-se essas informações mais facilmente. O utilitário *ddf* (*domain df*) foi implementado com este objetivo. De qualquer máquina pertencente ao domínio, tem-se acesso às informações dos discos de todas as máquinas, tornando automática a reunião dessas informações. Esse processo facilita o trabalho do administrador, pois dessa maneira ele poderá aproveitar de forma bem mais eficiente os discos e as informações referentes a esse recurso.

Além disso, o acréscimo de algumas opções não existentes ao comando df permite a visualização dos dados de uma maneira mais moderna e amigável.

2 Trabalhos Relacionados

2.1 df

O utilitário df possibilita ao administrador o controle do espaço em disco de uma determinada máquina, auxiliando na alocação de espaço no sistema.

df apresenta a quantidade de espaço em disco disponível no sistema de arquivos. Além dessa informação, df pode apresentar dados sobre *i-nodes*, ao invés de blocos utilizados, outro recurso que também pode se esgotar em sistemas de arquivos. Os dados são apresentados exibindo, para cada sistema de arquivos, o total de blocos (em *Kilobytes* ou 512 *bytes*) ou *i-nodes*, a quantidade e a porcentagem utilizadas, a quantidade disponível e o ponto de montagem da arquitetura.

2.2 mdf

O projeto *mdf* (*modern* df) [2], foi desenvolvido com o objetivo de facilitar o monitoramento dos *filesystems* no sistema operacional UNIX, especificamente no ambiente SunOS 4 e Solaris.

Uma das principais necessidades surgidas e que motivaram o desenvolvimento desta ferramenta é a incorporação de tecnologias que tornam a utilização dos sistemas mais segura, como a ferramenta *Undelete*, desenvolvida pelo projeto AHAND no Departamento de Computação da UNICAMP, que visa possibilitar a recuperação de arquivos acidentalmende apagados do sistema.

Esta ferramenta faz uma utilização eficiente do espaço livre nos *filesystems* para desempenhar suas tarefas.

Quando se utiliza o *Undelete* e o espaço livre indicado por df chega a zero, existe na realidade mais algum espaço que esta ferramenta pode ceder à medida que o sistema necessite. Seria então adequado saber qual o espaço que pode ser utilizado ainda.

Uma das funcionalidades do *mdf* é justamente informar a área dos *filesystems* que está potencialmente disponível.

Devido à existência de uma grande variedade de sistemas operacionais existentes, objetivou-se desenvolver um código que facilitasse a portabilidade para outros sistemas. Para isto, as rotinas foram separadas em dependentes e independentes do sistema utilizado.

Algumas outras necessidades foram atendidas pelo *mdf*. Entre elas destaca-se:

- a representação dos dados em unidades maiores como, megabytes e gigabytes.
- facilidade de visualização.

O código foi desenvolvido sem se basear nos códigos fontes existentes para alguns sistemas de domínio público, pois eles não apresentaram características de portabilidade como se desejava neste projeto.

As rotinas responsáveis pela coleta de dados, e que são dependentes do sistema utilizado, foram separadas das demais. Os dados recolhidos por estas rotinas são armazenados numa lista ligada para serem tratados por rotinas independentes do sistema.

No sistema operacional SunOS 4 utiliza-se a função `setmntent` para iniciar o processo de coleta de dados e `getmntent` para adquirir as informações dos *filesystems*. No sistema operacional Solaris utiliza-se a função `getmntent` para coletar os dados.

O acesso às informações referentes à utilização do *filesystem* pelo *Undelete* é feito através do comando `ud_query`, o qual é disponibilizado junto com a ferramenta *Undelete*.

3 Utilitário ddf

3.1 Arquitetura

O projeto é composto de *servidores df*, *centralizadores* e *clientes usuários*. Todas as máquinas pertencentes ao domínio podem ser servidores df. Periodicamente (o intervalo é definido pelo administrador, o *default* é 2 minutos), um processo no servidor df gera os arquivos <máquina>_df e i_<máquina>_df, com as respectivas entradas mdf -kls e mdf -ils. A opção -l apresenta somente os sistemas de arquivos locais, a opção -s mostra o tipo do sistema de arquivos, a opção -k exibe os blocos em *kilobytes* e a opção -i traz informações a respeito dos *i-nodes*. Na inicialização, o servidor df registra-se em um arquivo disponível via *NFS*: *hosts_df*. Somente as máquinas servidoras df serão alcançadas pelo utilitário ddf.

Na inicialização de um centralizador, o arquivo hosts_df é lido e são consideradas essas máquinas (servidores df) para as requisições. O centralizador também registra-se em outro arquivo disponível via *NFS*, *ddf.conf*, para uso do cliente usuário. O centralizador trabalha como cliente RPC dos servidores df. Periodicamente (o intervalo é definido pelo administrador, o *default* é 5 minutos), o centralizador faz, seqüencialmente, requisições RPC a todas as máquinas registradas em hosts_df e centraliza localmente em arquivos (esse processo é melhor detalhado na seção 3.2).

Pode haver mais de um centralizador para aumentar o grau de tolerância

a falhas. Tipicamente são em pequeno número (dois ou três) para manter baixo o tráfego na rede.

Os arquivos centralizados são importados pelo cliente usuário na execução do comando ddf, que está disponível via *NFS*, com o comando *remote copy*. Através do *header* (que será explicado na seção 3.2), determina-se qual arquivo é o válido. Em caso de mais de um centralizador, é iniciada a cópia remota em todos paralelamente (em *background*) e inicia-se o processamento com a primeira cópia a chegar. Os centralizadores são encontrados pelo arquivo ddf.conf. O *script* ddf filtra o arquivo segundo algumas opções do comando e é utilizado pelo usuário, que normalmente é o administrador da rede, para obter as informações de todos os discos pertencentes ao domínio. A figura 1 resume a arquitetura do utilitário ddf.

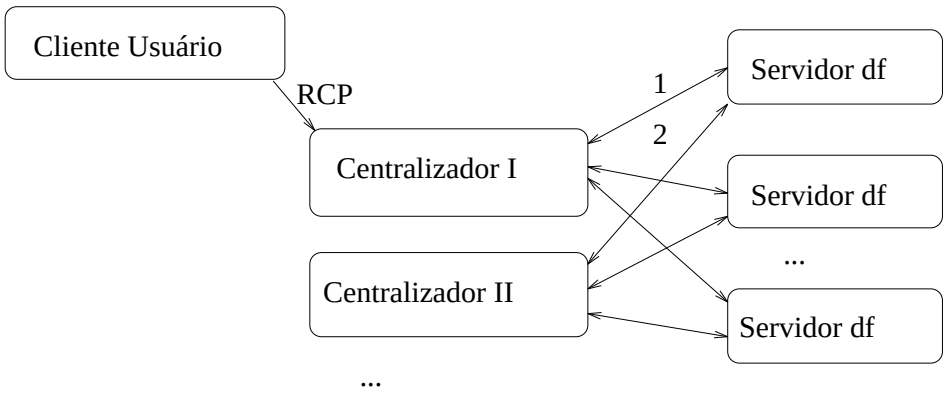

Figura 1: Arquitetura do utilitário ddf

A utilização de arquivos disponíveis via *NFS* restringe-se a arquivos de configuração e do comando ddf propriamente dito. A decisão quanto a disponibilidade dos arquivos via *NFS* ou não, foi influenciada pelo número de vezes que o arquivo é acessado. Arquivos muito acessados, se disponíveis via *NFS*, necessitariam ser montados freqüentemente, o que poderia criar muitas dependências no sistema. Por exemplo, se uma hierarquia de diretórios é montada muito freqüentemente, a queda de uma máquina responsável por um elemento dessa hierarquia pode inviabilizar a montagem de toda hierarquia.

A simples utilização de *NFS*, embora moderadamente, acaba restringindo sua utilização a sistemas em que todas estações compartilhem um disco via *NFS*. Utilizamos *NFS* visto que atualmente a maioria dos ambientes distribuídos dispõe dessa facilidade e seu uso vem se tornando cada vez mais freqüente. Observamos que em redes de médio ou grande porte a utilização de *NFS* é

quase universal, devido as facilidades geradas pelo seu uso. A possibilidade de, com o *automount*, gerar hierarquias virtuais sob demanda de diretórios distribuídos em várias máquinas como se fossem locais, faz do *NFS* uma ferramenta poderosa e quase imprescindível para a administração das redes atuais.

Apesar de uma arquitetura razoavelmente complexa, o uso do ddf é simples e fácil, podendo o administrador acessar todas as informações de qualquer máquina do domínio. Esta complexidade de projeto é necessária para evitar a criação de dependências de montagem *NFS* para os arquivos com as informações inerentes ao fornecimento do serviço.

3.2 Detalhes da Implementação

Atualmente, o utilitário ddf funciona nos sistemas SunOS e Solaris. É provável que sua extensão para outros sistemas UNIX seja simples. Foi implementado parte em *RPC* [3] e parte em KornShell [1], utilizando o compilador GCC. Este foi utilizado pois é de livre acesso e bastante confiável, além de amplamente documentado.

Para evitar detalhes de interface com a rede, utilizamos programação RPC. O paradigma RPC permite a construção de aplicações distribuídas em um nível de programação mais alto. Para facilitar ainda mais a programação dessas aplicações foi utilizado RPCgen, que gera *stubs* com as *templates* necessárias para a construção dos programas servidores e clientes RPC. Optamos por transporte orientado a conexão mas, devido a não existência de fatores críticos no projeto, poderíamos ter optado por datagramas (sem conexão).

A utilização de *RPC* também é interessante para tornar mais simples a manutenção do utilitário. A inclusão ou remoção de algum centralizador, por exemplo, é simples pelo fato dos servidores df serem passivos (esperam a requisição), bastando remover ou incluir o processo centralizador na máquina desejada, sem necessidade de reconfigurar os servidores df.

A centralização das informações contidas nos arquivos gerados pelos servidores_df é armazenada nos arquivos xyz@ddf_0 ou xyz@ddf_1 e i_xyz@ddf_0 ou i_xyz@ddf_1. Esses nomes foram escolhidos para evitar coincidências entre esses arquivos e outros existentes no sistema. O centralizador reveza a atualização dos arquivos 0 e 1, atualizando o arquivo *head_ddf* para definir o arquivo válido, conforme figura 2.

Esse processo é necessário, pois há concorrência entre as operações de atualização e leitura do arquivo centralizado. Sendo assim, o arquivo com as

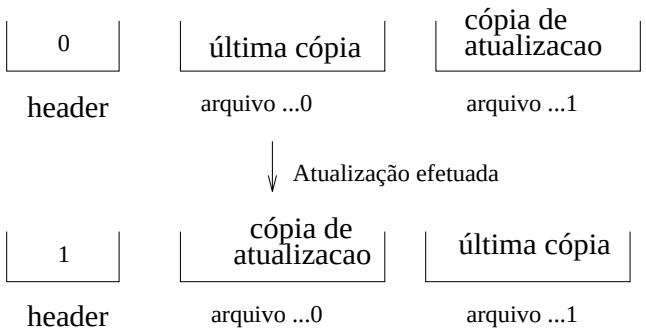

Figura 2: Atualização do arquivo centralizado

informações centralizadas pode ser requisitado durante sua atualização. Para que essa requisição seja consistente, o cliente usuário considera válido o arquivo indicado pelo *header*, sendo o outro considerado uma cópia de atualização. Quando é terminada a atualização, a cópia de atualização passa a ser a cópia válida e vice-versa através de uma alteração no *header*. Esse processo é conhecido como *shadow copy* e já é utilizado para garantir consistência de transações [4].

Durante a ativação do comando ddf, os arquivos dos centralizadores são importados em paralelo. Para evitar sobreposição desses arquivos, é acrescentado como prefixo o nome do centralizador de origem do arquivo.

O projeto verifica uma série de condições para tornar realmente interessante seu uso. Por exemplo, se a criação do arquivo de configuração (ddf.conf) falhar, por ocasião do registro do centralizador por qualquer motivo (servidor *NFS* caído...), o centralizador insiste de cinco em cinco minutos na operação. Se depois de um dia não obtiver sucesso, um mail é enviado ao administrador para alertá-lo do problema.

Para facilitar a desativação do ddf, também é disponibilizado um *script* que derruba todos os processos e remove todos os arquivos referentes ao utilitário.

3.3 Utilização

O comando ddf tem o seguinte sintaxe:

ddf [-h <lista de hosts>] [-t] [-a] [-K <lista de palavras chaves>] [-k] [-m] [-g] [-i]

Opções:

- -h, exibe somente os *filesystems* dos *hosts* especificados na lista;
- -t, lista todos os *filesystems* locais, inclusive os montados localmente;
- -a, ordena os arquivos por ordem de espaço livre, segundo o campo *available*;
- -K, exibe os *filesystems* que possuam o conjunto de palavras chaves especificado;
- -k, apresenta o espaço em *kilobytes*;
- -m, apresenta o espaço em *megabytes*;
- -g, apresenta o espaço em *gigabytes*;
- -i, apresenta a utilização de *i-nodes*.

Os argumentos do comando ddf funcionam da esquerda para a direita, com exceção do argumento -t que tem mais alta prioridade. Para compreender a saída gerada basta percorrer logicamente o conjunto de argumentos (a ordem dos argumentos pode alterar a saída).

4 Conclusões e Possíveis Extensões

O utilitário ddf descrito aqui vem facilitar o acesso a informações importantes sobre discos. Estas informações eram de trabalhoso acesso, necessitando ao administrador obtê-las máquina a máquina.

O projeto do ddf também se preocupa com a facilidade e confiabilidade da instalação e manutenção do software no ambiente, visto que o administrador já é bastante ocupado com a administração da rede e não deve ter trabalho extra com instalação e ajustes complexos ou trabalhosos de ferramentas, assim como com a monitoração mandatória de discos hospedeiros de *homedirs*.

Ainda não foi efetuada uma análise dos efeitos do utilitário no sistema como um todo (tráfego, carga em CPU, ...). Em princípio, podemos deduzir que o tráfego do ddf não degrada significativamente o desempenho do tráfego geral, visto que ele não é constante (ocorre periodicamente) e o número de participantes é O(nN), onde n é o número de centralizadores e N é o número de máquinas. No pior caso (número de centralizadores igual ao número de máquinas), será $O(N^2)$. Os arquivos centralizados são pequenos e o número de centralizadores também costuma ser. Além desse tráfego, há outro durante a execução do ddf (importação dos arquivos centralizados), mas sua freqüência típica deve ser pequena. Os *daemons* do sistema também não

consomem significativamente a CPU. Entretanto, um estudo mais completo e detalhado do efeito do ddf no desempenho da rede ainda é necessário para dados mais precisos.

A análise do utilitário quanto a sua utilidade e facilidade de uso, através de sua utilização por administradores em ambiente de trabalho, poderá ser interessante e de grande ajuda para o aprimoramento do utilitário, visando melhor satisfazer as necessidades reais existentes na administração do sistema. No momento, sua instalação no domínio dcc.unicamp.br está em andamento.

O utilitário ddf foi desenvolvido por Marco Aurélio Medina de Oliveira e Vitor Hugo Furtado sob orientação do professor doutor Paulo Lício de Geus como resultado da disciplina Tópicos em Redes de Computadores, ministrada pelo mesmo. Da mesma forma, o utilitário mdf foi desenvolvido anteriormente por Carlos Fernando Bella Cruz.

O sentido desses projetos é dar continuidade aos trabalhos resultantes da disciplina, visando formar assim um conjunto de ferramentas para administração de redes.

Referências

[1] Morris I. Bolsky and David G. Korn. *The Kornshell Command and Programming Language*. Prentice Hall, 1989.

[2] Carlos Fernando Bella Cruz. Man page do utilitário Mdf no domínio dcc.unicamp.br. 1994.

[3] *Network Interfaces Programmer's Guide. SunOS 5.2 Answerbook*, May 1993.

[4] Graham D. Parrington, Santosh K. Shrivastava, Stuart M. Wheater, and Mark C. Little. *The Arjuna System Programmer's Guide*. Department of Computing Science, The University, New Castle Upon Tyne, 1995.