ESCOLA POLITÉCNICA
DE PERNAMBUCO

UPE - POLI
Engenharia de Computação
GRADUAÇÃO

ESTUDO DE VIABILIDADE COM ENFOQUE EXPERIMENTAL PARA A IMPLANTAÇÃO DE SISTEMA DE *BACKUP OPEN SOURCE* EM AMBIENTE CORPORATIVO

Trabalho de Conclusão de Curso

Engenharia da Computação

Itagiba C. Carneiro Leão

Orientador: Profa. Maria Lencastre

UNIVERSIDADE
DE PERNAMBUCO

ITAGIBA C. CARNEIRO LEÃO

ESTUDO DE VIABILIDADE COM ENFOQUE EXPERIMENTAL PARA A IMPLANTAÇÃO DE SISTEMA DE *BACKUP OPEN SOURCE* EM AMBIENTE CORPORATIVO

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia da Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Recife, Dezembro e 2010.

A todos que me ajudaram neste trabalho.

Agradecimentos

Agradeço a Deus, à minha família, à minha orientadora, professora Maria Lencastre e a todos que de alguma forma contribuíram para a realização deste trabalho.

Resumo

A demanda de dados cada vez maior e a necessidade de segurança das informações requer a elaboração de um projeto de arquitetura de *backup* com dimensionamento dos recursos tecnológicos e de pessoal habilitado. Hoje existem várias soluções de sistemas de *backup* corporativos velozes, flexíveis e confiáveis por meio de empresas privadas especialistas no assunto. Contudo, há soluções em *softwares* livres que também propiciam um bom gerenciamento em implementações de *backup* de igual confiabilidade e eficiência, diminuindo os custos com a aquisição de *softwares* proprietários. O objetivo do projeto é realizar um estudo da viabilidade com enfoque experimental para a implantação de sistemas de *backup/restore open source* em empresas, utilizando o *software* Bacula e os dispositivos de *hardware* disponíveis, tais como as bibliotecas de fitas SUN StorageTek L20 e IBM TS3200, verificando a compatibilidade entre *software* e *hardware*, além de comprovar as funcionalidades básicas de *backups e restores* em um ambiente corporativo.

Abstract

The increasing demand for data and the need for information security require the elaboration of a backup architecture project with scaling up of technological resources and trained personnel. Today there are several solutions to enterprise backup systems fast, flexible and reliable by private specialists. However, there are solutions in free software that also offer proper management of backup implementations with the same reliability and efficiency, reducing costs with the acquisition of proprietary software. The project goal is to conduct a feasibility study with an experimental approach to the deployment of open source backup/restore systems in companies, using the Bacula software and available hardware devices, such as tape libraries Sun StorageTek L20 and IBM TS3200, verifying the compatibility between software and hardware, and demonstrate the basic features of backups and restores in a corporate environment.

Sumário

Índice de Figuras

Índice de Tabelas

Tabela de Símbolos e Siglas

Em ordem alfabética:

- **CD-ROM:** *Compact Disc Read-Only Memory*
- **CPU***: Central Processing Unit*
- **D2D:** *Disc to Disc*
- **D2T:** *Disc to Tape*
- **DVD:** *Digital Video Disc*
- **FC:** *Fibre Channel*
- **GB:** *Gigabyte*
- **Gbps:** *Gigabits per second*
- **GNU:** GNU's *Not Unix*
- **GPL:** Licença Pública Geral
- **GUI:** *Graphical User Interface*
- **HD:** *Hard Drive*
- **IP:** *Internet Protocol*
- **LAN:** *Local Area Network*
- **LTO:** *Linear Tape Open*
- **LUN:** *Logical Unit Number*
- **L2TP:** *Layer Two Tunneling Protocol*
- **MB:** *Megabyte*
- **MB/s:** *Megabytes per second*
- **Mbps:** *Megabits per second*
- **NAS:** *Network-attached storage*
- **RAID:** *Redundant Array of Inexpensive/Independent Drives*
- **RAM***: Random Access Memory*
- **SAN:** *Storage Area Network*
- **SCSI:** *Small Computer System Interface*
- **SGBD:** Sistema de Gerenciamento de Banco de Dados
- **SO:** *Sistema Operacional*
- **TCC**: Trabalho de Conclusão de Curso
- **TCP/IP***: Transmission Control Protocol/Internet Protocol*
- **TI:** Tecnologia da Informação
- **TLS:** *Transport Layer Security*
- **VPN**: *Virtual Private Network*
- **WAN:** *Wide Area Network*

Capítulo 1
Introdução

O *backup* (ou cópia de segurança, em português) está relacionado com a cópia de dados de um dispositivo de armazenamento para outro, permitindo que os mesmos possam ser restaurados em caso de perda dos dados originais como, por exemplo, quando ocorrem remoções acidentais ou corrupção de dados [6]. Bons administradores de sistema sabem que implementar um procedimento robusto de *backup* é uma das mais importantes tarefas, e é também uma das mais complexas. Pode-se dizer que *backup* está relacionado à recuperação, assim como recuperação está relacionado à confiabilidade e proteção. Falhar na restauração de arquivos pode representar a ruína para uma empresa e seus clientes, além de toda uma carreira promissora de um administrador de sistemas. Logo, ter uma ferramenta de *backup* confiável é uma questão essencial para uma empresa.

A escolha por um *software* de *backup* corporativo que melhor se adeque às necessidades das empresas, de modo eficaz e usando os recursos de *software*/*hardware* disponíveis, é quase sempre um desafio. Esta escolha envolve estudos de viabilidade que verifiquem a compatibilidade entre todos os componentes da arquitetura de *backup*, desde diferentes distribuições e versões de sistemas operacionais (SOs), com dependências de pacotes distintos, até *drivers* de dispositivos. Além disso, em ambientes corporativos as soluções de *backup* geralmente requerem usuários com conhecimentos avançados em administração de sistemas, capazes de configurar uma ferramenta de *backup* que esteja preparada para qualquer eventualidade. Com uma ferramenta de *backup* vêm também uma série de funcionalidades e, a depender da política de *backup* da empresa, pode ser necessário realizar implantações mais específicas como: esquemas de agendamento de *backup*, *backups* locais, e *backups* remotos.

Hoje em dia existem várias soluções de sistemas de *backup* corporativos velozes, flexíveis e confiáveis, disponibilizados por empresas privadas e especialistas no assunto [1] [2]. Algumas soluções requerem poucos recursos e baixo custo, enquanto outras exigem muitos recursos e são caras. Contudo, há soluções de *softwares* livres que também propiciam um bom gerenciamento em implementações de *backup*, de igual confiabilidade e eficiência às soluções de empresas privadas, diminuindo os custos com a aquisição de *software* proprietário. Além disso, o direito ao livre acesso ao código fonte tem despertado o interesse para os *softwares* de *backup open source*, no que se refere à personalização e adequação dos mesmos aos requisitos exigidos pelas empresas.

Diversas iniciativas têm surgido no Brasil motivando o uso de *software* livre. A Comissão de Ciência e Tecnologia, Comunicação e Informática da Câmara dos Deputados aprovou a proposta que garante preferência para *software* livres na contratação de bens e serviços de informática pela União, pelos estados, pelo Distrito Federal e os municípios [32]. Segundo a deputada Luiza Erundina (PSB-SP) [32] “Estima-se que o Estado, em todos os seus níveis, gaste cerca de 2 bilhões de dólares por ano com pagamento de aluguel de licenças

de programas-proprietários". Ainda segundo [32], a lei obrigará a administração a adotar obrigatoriamente a licitação do tipo "técnica e preço" para a contratação de bens e serviços de informática, e que a contratação de programas-proprietários só ocorrerá no caso de "justificada inadequação" do *software* livre. Neste caso, a avaliação das propostas deverá considerar os custos totais, incluindo instalação, licenciamento, instalação e suporte. Por outro lado existem dados relevantes sobre o *software* proprietário. Segundo o pregão eletrônico [33], Agencia Nacional de Vigilância Sanitária (Anvisa), adquiriu licenças e manutenção do *software* IBM Tivoli Storage Manager (TSM), no valor de R$ 480.020,02 para o período de mar./2006 à mar./2007. Já no pregão eletrônico n°26/2010, [34], o valor de atualização do TSM versão 5.5 para a versão mais atual por um período de 24 meses, custará cerca de R$ 148.884,80 à Controladoria-Geral da União (CGU).

Este trabalho foi motivado por um problema encontrado durante a realização de um estágio numa empresa pública de médio porte, que possuía dificuldades com o sistema de *backup* em uso, pois o mesmo já se encontrava obsoleto (devido à falta de atualização), e havia demora na aquisição de um novo sistema proprietário mais poderoso e atual. O objetivo foi então encontrar um *software de backup* que atendesse bem aos requisitos de *backups* corporativos mais comuns, se adequando às condições de ambiente operacional pré-existente nessa empresa. Através de pesquisas realizadas na internet, optou-se por realizar um estudo com o *software* Bacula, que mostrou ser de grande popularidade, dentre uma gama de opiniões de usuários e empresas que o utilizam, bem como segundo as estatísticas do site Sourceforge [3]. O Bacula é um sistema de *backup open source* de nível corporativo para ambientes com demanda de recursos de *software/hardware* heterogêneos [4]. Sua arquitetura modular permite o gerenciamento de *backup*, recuperação e verificação de dados de maneira distribuída ao longo de uma rede, adotando um modelo cliente/servidor. Estes módulos trabalham juntos para fornecer uma solução de *backup* robusta e completa para ambientes com vários sistemas operacionais. Transpor a barreira de compatibilidade de *software* e *hardware*, neste estudo de caso de ambiente corporativo, em particular, passou a ser o objetivo principal no estudo da viabilidade da implantação do Bacula, além de comprovar suas funcionalidades básicas de *backup/restore*.

1.1. Estratégia Adotada

A estratégia empregada para a execução e viabilização deste projeto incluiu:

- Identificação de um software de backup corporativo *open source* a ser estudado como referência para realizar o estudo de viabilidade dentro da empresa.
- Definição de conjunto mínimo de requisitos a serem atendidos pelo sistema analisado na empresa, como por exemplo: gerenciamento centralizado, restauração e verificação de dados e compartilhamento de recursos de rede com os demais aplicativos, com boa vazão, assegurando a integridade e confiabilidade dos dados copiados dos clientes para suas oportunas restaurações.
- Estudo do *software* Bacula usando os dispositivos de *hardware* disponíveis.

Muitos fóruns e *sites* foram visitados, além do próprio *site* do Bacula, com o intuito de obter não apenas os pacotes de softwares e bibliotecas requisitadas, mas também dicas de uma configuração de instalação mais apropriada. Há divergências no que realmente é necessário na instalação, uma vez que cada um tem suas preferências e prioridades. O manual do Bacula, também deixou algumas lacunas, que aos poucos foram vencidas. Não foram encontradas condições semelhantes, com as mesmas necessidades e que a versão do Bacula utilizada fosse a mais recente. Instalado no SO CentOS 5.5, arquiteturas de 32 *bits* e 64 *bits*, a intenção foi observar a sua funcionalidade usando como recurso de *hardware* dois ambientes distintos.

O primeiro passo para uso do Bacula, assim como descreve o manual [5], foi verificar e atender aos requisitos de *software* e *hardware* para a sua instalação e fazer o teste usando o utilitário ***btape*****,** disponível também no *site*, a fim de checar a funcionalidade da biblioteca de fitas (dispositivo de fita – indicada como meio para guardar os dados em empresas).

- Na fase de desenvolvimento prático foram realizadas as etapas necessárias às fases de instalação e configuração. O sistema foi então testado frente às funcionalidades básicas cotidianas. Por meio da implantação de um protótipo, a viabilidade do projeto foi verificada com base na comparação entre os resultados obtidos e os resultados esperados, sendo considerados 2 ambientes.

1.2. Estrutura do Documento

Este documento está organizado da seguinte forma:

No Capítulo 1 é feita a contextualização do problema de *software* de *backup* corporativo, juntamente com a motivação e objetivo do trabalho realizado.

No Capítulo 2, é detalhada a importância do *backup* dentro do contexto corporativo. e são descritos os principais conceitos de *backup*.

No Capítulo 3 são abordados os tipos de arquitetura de rede que compõem a infraestutura de *backup*.

No Capítulo 4 é apresentado um comparativo técnico entre o *software* de *backup* Bacula em relação a algumas ferramentas presentes no mercado.

No Capítulo 5 são abordados os principais conceitos básicos para a compreensão do *software* Bacula.

No Capítulo 6 são delimitados os ambientes de teste e de implantação, visando o projeto da arquitetura do sistema.

No Capítulo 7 é apresentada a Conclusão e os trabalhos futuros.

Capítulo 2
Conceitos de Sistema de *Backup*/*Restore* Corporativo

2.1. Importância do *Backup*

O rápido crescimento das informações nas infraestruturas de TI requer eficiência do sistema de *backup* com o objetivo de manter a segurança de recursos importantes. Todas as empresas devem ter um sistema de *backup* como requisito de negócio, para não pôr todo investimento e credibilidade em risco.

A perda de dados pode ocorrer devido a falhas de dispositivos de *hardware*, erros humanos, roubos, invasões, vírus, mau funcionamento de *softwares*, e até mesmo a desastres naturais, como enchentes. Para esses casos serem evitados, os dados devem ser copiados e armazenados fora do local (*off-site*), como parte de um plano qualificado como *Disaster Recovery*.

A maioria dos pedidos de recuperação dos dados ocorre devido a erros humanos. Nesses casos, devido a mais rápida capacidade de acesso aleatório em armazenamento em disco, a recuperação de disco rígido local é bem mais rápida do que recuperação em fitas.

Para o usuário comum, o *backup* se restringe a algumas limitações, e cópias de uma pasta e/ou arquivos para um HD, CD-ROM, pen-drive ou DVD feitas facilmente com rapidez. Já em empresas, a arquitetura de ambiente computacinal possui maior complexidade, variando de acordo com o tamanho e/ou estrutura organizacional. Desta forma, se faz necessário uma solução mais arrojada através de um sistema de *backup* corporativo, que permita toda interatividade e automatização na realização das tarefas rotineiras de *backup*, de modo seguro, confiável e centralizado.

2.2. Principais Requisitos de um Sistema de *Backup* Corporativo

- **Gerenciamento centralizado:** permite a execução de todas tarefas tradicionais de *backup/restore* através de uma console central. A maioria das empresas de gerenciamento de armazenamento, como a IBM, [1], e a Computer Associates International Inc., [2], oferecem opções para este processo.

- **Automação das tarefas de gerenciamento de armazenamento e recuperação:** automação reduz custos operacionais, os erros de mídia, e permite executar rotinas de proteção de dados centralizadamente.

- **Métodos de verificação como checagem dos dados na hora do armazenamento:** assegura a integridade dos dados guardados.

- **Suporte a ambientes heterogeneos:** permite o gerenciamento de *backup*, recuperação e verificaçao dos dados de vários servidores rodando em várias plataformas numa rede de computadores.

- **Compressão dos dados armazenados:** acelera a transferência de dados, reduzindo o tráfego de rede.

- **Execução de comandos ou rotinas antes e depois dos *jobs*:** no caso de um *job* falhar, por exemplo, uma mensagem pode ser enviada por email ou celular.

- **Suporte à bibliotecas de fitas de um simples *drive* à bibliotecas com múltiplos *drives* e *slots*.**

- **Criptografia dos dados a serem salvos:** assegura que os dados sejam trafegados na rede de forma segura sem que sejam capturados por intrusão maliciosa ou acidental. Técnicas de encriptação podem ser utilizadas mas degradam a perfrmance do *backup* e do *restore*.

- **Deduplicação de dados:** trata-se de uma tecnologia nova que checa a redundância dos dados que serão "backupeados" ou arquivados, tanto para *storages* em disco quanto em fita. Se baseia, no nível de arquivos, em salvar uma única cópia dos arquivos idênticos e substitui todas as outras por marcadores que apontam para essa cópia; ou no nível dos sub-arquivos, quando realiza uma varredura nos blocos de dados um a um, identifica os bytes idênticos, e os substitui por indicadores, resultando em uma significativa redução das exigências quanto à capacidade de armazenamento. A taxa de redução do *backup* vai depender do tipo de dado, dos períodos de retenção, da frequencia de *backups* completos e da tecnologia de deduplicação adotada.

- **Solução escalável:** é a base para assegurar a integração de produtos para proteção de dados, disponibilidade, migração, arquivamento para cumprimento de normas, recursos de *storage* e gerenciamento de SAN, por meio de uma plataforma automatizada com catálogo e índice globais.

- **Documentação atualizada e disponível.**

- **Suporte técnico**.

2.3. Tipos de *Backup*

- ***Backup* Completo, Full, total ou normal**: faz o *backup* na íntegra de todos arquivos e diretórios selecionados para a mídia de *backup*. Cada arquivo "backupeado" tem seu atributo de arquivo desmarcado (*bit* de *archive* = *off*). Consome maior capacidade de mídia e, portanto, devendo ser realizado em situações de mudanças frequentes de

dados. Deve ser feito em intervalos de tempo maiores. Inicia o ciclo dos *backups* diferencial e incremental.

- ***Backup* Diferencial**: é um *backup* cumulativo de todos arquivos criados ou alterados desde o último *backup* completo. O atributo de arquivo não é modificado (*bit* de *archive = on*). *Backup* normalmente longo e de *restore* rápido. Juntos, um *backup* completo e um *backup* diferencial devem incluir todos os arquivos no computador, alterados e inalterados. Uma vez por semana é considerado o intervalo ideal entre uma cópia diferencial e outra. A restauração de arquivos e pastas exigirá o último *backup* completo e o último *backup* diferencial.

- ***Backup* Incremental:** é feito o *backup* apenas dos arquivos criados ou alterados desde o último *backup* completo ou incremental. O atributo de arquivo é desmarcado (*bit* de *archive = off*). Backups incrementais são mais rápido que os *backups* completos ou diferenciais. O *backup* dos dados que utiliza uma combinação de *backups* completo e incremental exige menos espaço de armazenamento e é o método mais rápido para *backup*. No entanto, a recuperação de arquivos pode ser difícil e lenta, porque o conjunto de *backup* pode estar armazenado em vários discos ou fitas.

Ao contrário dos *backups* completos, os *backups* incrementais primeiro verificam se o horário de alteração de um arquivo é mais recente que o horário de seu último *backup*. Se a data de modificação é mais recente que a data do último *backup*, o arquivo foi modificado e deve ter seu *backup* feito.

2.4. Política de *Backup*

A Política de *Backup* compreende a análise, seleção e organização dos dados que deverão ser assegurados segundo suas peculiaridades de requisitos de negócio e legais, estabelecendo-se estratégias para a devida realização do *backup* em tempo hábil, a fim de minimizar a perda das informações. É uma tarefa muito complexa, embora pareça simples. Em primeira instância, deve-se analisar quais informações mais importantes dentro da empresa, atribuir uma escala de prioridades, ter conhecimento da frequência de suas atualizações, verificar tamanho e taxas de crescimento de arquivos, disponibilidade de recursos, tempo de vida dos dados.

Arquivos de sistemas dos servidores: configuração, bancos, *logs*, por exemplo, devem ter uma prioridade alta na política de *backup*, evitando a reconstrução de todo um sistema por perdas parciais dos dados. Sistemas operacionais não precisam de *backup* em bases regulares. A estratégia de *backup* é condizente com a natureza dos dados armazenados. Este processo envolve todos os responsáveis pelos setores da empresa, pois estão mais aptos a prestar as informações necessárias para construção da política de *backup*. As informações são analisadas e organizadas da melhor maneira pelo administrador de *backup*, que define as estratégias mais adequadas, em geral, para serem automatizadas no processo de *backup*, através de *scripts* ou *jobs*. É preciso monitoração e avalição constantes para melhor viabilizar o sistema. Uma vez estruturada a política de *backup*, as informações, abaixo descritas, são imprescindíveis para sua efetivação.

- Identificação dos servidores - IP e/ou *hostname*.
- Especificação da(s) pasta(s)/arquivo(s) e seus respectivos caminhos (*PATH*).
- Período de retenção - tempo em que os dados ficam guardados, após este período, são descartados, liberando a área/midia de armazenamento para reaproveitamento.
- Periodicidade ou frequência de armazenamento: se diário, semanal, mensal, anual ou misto. Depende do volume de alterações e criações de arquivos feitas no decorrer de um dia, uma semana, por exemplo.
- Qual o tipo: completo, incremental ou diferencial.

2.5. RAID não é *Backup*

A redundância de discos rígidos fornecida pelo RAID, [7], não é uma solução de *backup*. Apesar do RAID oferecer segurança e confiabilidade na adição de redundância e evitar alguns tipos de falhas de discos, o RAID não protege contra falhas de energia ou erros de operação, [8]. Segundo, [9], protege apenas contra um tipo de perda de dados: a falha de um único disco rígido. De forma geral, o RAID foi criado para aumentar a velocidade de acesso a arquivos (processamento de dados), e aumentar o tamanho lógico de uma unidade de disco rígido. RAID é um termo usado para esquemas de armazenamento de dados que dividem e/ou replicam dados entre vários discos rígidos. Permite o uso de múltiplos *drives* (um *array* de *drives*) para aumentar o desempenho e/ou confiabilidade do *drive*. Para gerenciar a tecnologia RAID apropriadamente deve-se adquirir um servidor dedicado que suporta discos rígidos *hot-swap* e um controlador RAID SCSI. O servidor, o controlador RAID e o *firmware* dos discos rígidos necessitarão de atualizações periódicas, e, eventualmente, poderão ser substituídos quando mudar os padrões de tecnologia. Tudo isso terá que ser mantido por uma equipe especializada.

Os discos rígidos podem falhar, e se verificações de consistência não forem executadas regularmente no array RAID, pode ser impossível reconstruir a unidade com falha. Falhas no controlador RAID, na CPU, ou até mesmo nas placas de memória RAM, podem deixar, rapidamente, uma série de dados corrompidos.

Há muitos níveis de RAID que podem ser implementados usando *software* ou *hardware*. Muitas placas-mãe modernas incluem suporte ao *hardware* RAID. Níveis de RAID, como por exemplo, RAID 1 (conjunto espelhado) diminuem a possibilidade de perda de dados pela duplicação destes em todas unidades do array. Se um arquivo for corrompido ou perdido devido a uma falha do *drive*, pode-se recuperá-lo no *drive* espelho. Tudo isto é feito automaticamente sem a intervenção do usuário. O ponto importante a notar, contudo, é que, embora a possibilidade de perda de dados devido à falha de disco seja reduzida, a perda de dados é certa devido a vírus ou erros do usuário, por exemplo, a substituição ou exclusão de um arquivo. Independentemente do nível de RAID adotado, o *backup* é necessário.

Capítulo 3
Infraestrutura de *Backup*

A infraestrutura do *backup* corporativo deve ser criada para ser mais adaptativa possível às mais diferentes realidades, bem como, ser reconfigurada continuamente às necessidades dos servidores, volumes e novas tecnologias. Por isto é importante a elaboração, de fato, de um projeto de arquitetura de *backup* e não somente o dimensionamento e aquisição de recursos.

Fazem parte da infraestrutura de *backup*: o servidor de *backup*, biblioteca de fitas, *drives*, mídias, interfaces de rede, rede de dados, o *storage* de *backup* e inclusive o *hardware* dos clientes, além do sistema operacional, do *software* de *backup*, e, demais utilitários que podem fazer ou não parte do pacote de *backup*.

3.1. Arquiteturas de *Backup*

Uma arquitetura de *backup* é criada com a finalidade de realizar *backup* dos sistemas em produção disponibilizando-os com rapidez e confiabilidade no caso de uma necessidade de recuperação dos dados armazenados. Para tanto, deve ser considerado na elaboração de um projeto de arquitetura de *backup* fatores técnicos, estruturais, organizacionais e financeiros disponibilizados pela empresa para o correto dimensionamento dos recursos. Esta tarefa exige do administrador de *backup* muito empenho em se tratando de uma tarefa muito complexa pela heterogeneidade dos recursos de *software* e *hardware* envolvidos. Estes devem ser combinados ou integrados de maneira que possibilitem a utilização de sua máxima capacidade, garantindo a capacidade de vazão necessária para que os volumes de *backups* sejam feitos dentro das janelas definidas, bem como, ofereçam flexibilidade de adaptação a condições não planejadas.

A vazão real dos dados obtém-se percorrendo a cadeia dos componentes dimensionados no projeto de arquitetura, iniciando, por exemplo, no disco do cliente, passando pelo barramento, interface de rede, rede de dados, interface do servidor de *backup*, CPU, memória, barramento, interface de *storage*, biblioteca de fitas, *drives* e fitas. A melhor taxa de velocidade vai ser determinada pelo componente de pior desempenho. Isto significa também que em uma mesma instalação pode-se ter números diferentes em função do cliente. Isto justifica porque as vazões reais são menores do que as especificadas pelos fabricantes dos equipamentos. Outro fator importante a considerar é a existência de pontos de estrangulamento do fluxo de dados. No *backup* corporativo, além da questão da performance, tem-se que administrar com extrema atenção a competição por recursos.

O ciclo de atualização dos dados de cada aplicação determina a criação de uma janela de *backup* (faixa de tempo para fazer o *backup*). Um parâmetro crítico do *backup* é a taxa de transferência que é dada pela divisão entre quantidade de dados para *backup* e a janela de

backup . Por exemplo, para um *backup* de 148GB de dados com uma janela de 8 horas, tem-se 18.5GB/hora de taxa de transferência necessária.

Cada aplicação tem um determinado ciclo de operação e funcionamento que determina o período onde os seus dados são consultados e atualizados.

O *backup*, bem programado, deve ser posicionado para capturar e preservar um volume de atualizações que represente a menor perda aceitável. Assim, não existe um conceito absoluto de quando um *backup* deve ser realizado. Em geral, a janela de *backup* é resevada para a noite/madrugada, quando se verifica a menor demanda de uso da rede.

Segundo, [10],as principais arquiteturas de *backup* estão descritas abaixo.

3.1.1. LAN *Backup*

É uma arquitetura de *backup* em LAN que é implementada comumente por pequenas empresas, degradando o tempo de resposta para os usuários. Este *backup* é normalmente centralizado e automatizado com o uso de biblioteca de fitas baseada na tecnologia LTO. Em geral, usa como recursos servidor de *backup* conectado à biblioteca de fitas e à LAN. Nesta arquitetura, dados que estão contidos em *storage on-line* e acessados através de servidores de arquivos ou servidores de aplicação, como servidores de banco de dados, por exemplo, são puxados através de uma LAN Ethernet por um servidor de *backup* para a biblioteca de fitas a qual esta conectado.

A Figura 1 mostra vários elementos que podem potencialmente atuar como pontos de estrangulamento do desempenho deste sistema de *backup*. *Backups* tendem a diminuir o desempenho global da LAN, devido à grande quantidade de dados que flui através dela de uma vez. O resultado é uma redução no desempenho para os usuários no segmento de rede ao qual o sistema de *backup* em fita está conectada - reduzindo a eficiência global do sistema. Os próprios servidores representam um gargalo, especialmente servidores de arquivos. A consequência da redução do desempenho no servidor de arquivos ou de aplicação é tornar mais lento o desempenho global dos usuários que tentam acessar informações através do servidor. Não raro o servidor de arquivos pode não atender à pedidos de usuários finais durante a janela de *backup*. Finalmente, a relativa lentidão das bibliotecas de fitas também representa outro ponto de gargalo, embora pequeno no desempenho geral do sistema.

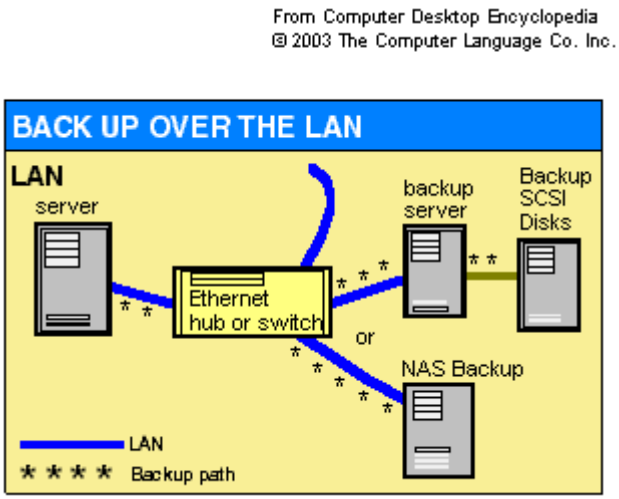


Figura 1. LAN *Backup*, [10].

Para melhorar o desempenho, pode-se definir uma rede independente para o *backup*. Abaixo a Figura 2 ilustra esta opção.

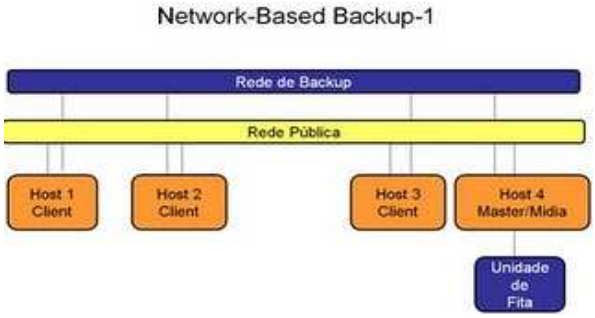


Figura 2. LAN *Backup* com rede de *backup* independente, [10].

Com o advento da tecnologia FC a implementação de novas arquiteturas de *backup* LAN-*free Backup* e *Server-less Backup* foram viabilizadas com vantagens significativas: aumento da performance do sistema, eliminação de pontos de gargalo do sistema, dimiuição do custo total de recursos. Estas vantagens beneficiam não só os administradores de sistemas/*backup*, engenheiros de rede, mas, principalmente, os usuários que dependem desses sistemas.

3.1.2. LAN-*free Backup*

Refere-se ao *backup* de dados, sem transferí-lo através da LAN ou WAN. A arquitetura mais simples LAN-*free* é um servidor ou dispositivo de armazenamento NAS diretamente ligado à uma biblioteca de fitas, Figura 3.

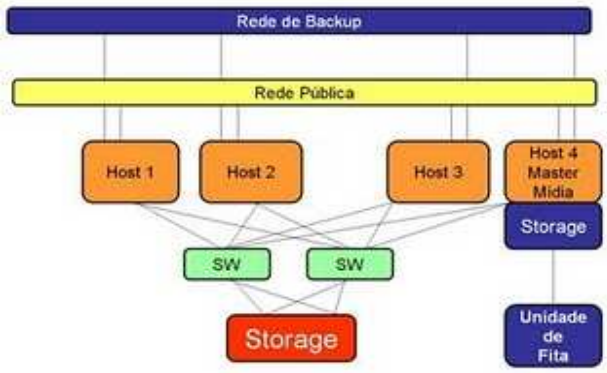


Figura 3. LAN-*free Backup*, [10].

Uma abordagem comum é utilizar uma SAN, que liga *storage* de discos, servidor de *backup* e biblioteca de fitas através de um *switch* FC. A operação de *backup* é gerenciada pelo servidor de *backup*, Figura 4.

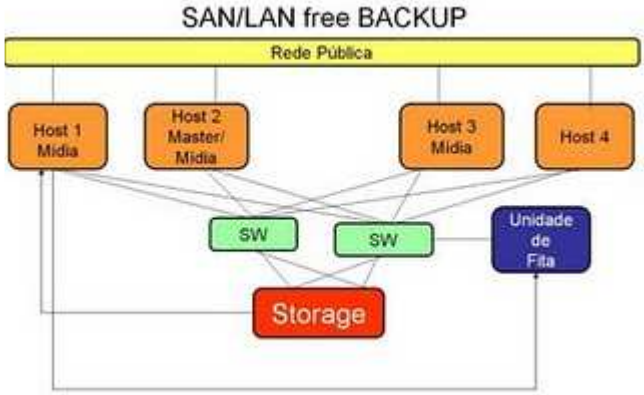


Figura 4. SAN/LAN-*free Backup*, [10].

A arquitetura SAN/LAN-*free Backup* normalmente é utilizada para *backups* de aplicações que manipulam grandes quantidades de dados em ambientes 24x7. Para funcionar exige uma rede de *storage* centralizado e permite um *restore* rápido.

Uma variação de LAN-*free* é o *Server-less Backup* (*backup* sem servidor). *Server-less Backup* usa uma SAN, mas em vez de ter um servidor de *backup* gerenciando a operação de *backup*, usa um dispositivo chamado "data mover." Este pode estar incorporado na *storage* de *backup* ou ser uma unidade separada que também lida com a conversão de FC para SCSI, Figura 5.

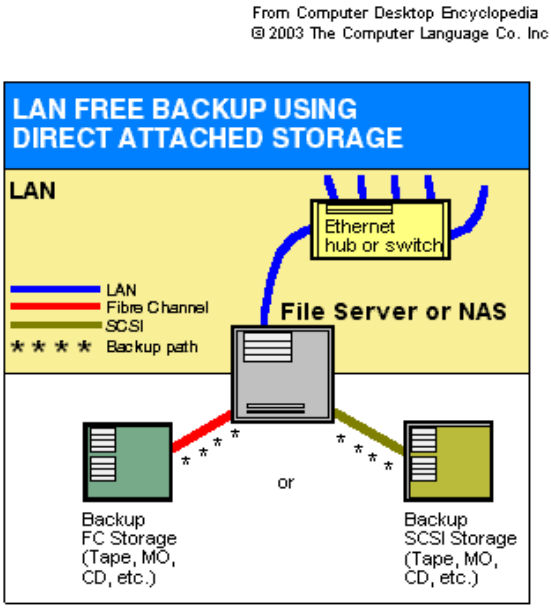


Figura 5. Variação de LAN-*free Backup*, [10].

Se os servidores e dispositivos de armazenamento NAS estão ligados diretamente à *storage* de *backup*, a rede em si não é afetada, mas os servidores e dispositivos NAS são. Eles têm de executar operações de *backup* e atender aos usuários, ao mesmo tempo. Uma SAN fornece uma solução de alta velocidade para fazer *backup* dos dados. Conexões FC entre os discos e a *storage* de *backup* são muito rápidas, na ordem de Gbps. Os dados são organizados freqüentemente em áreas separadas no disco inicialmente, para que os usuários e *backup* não trabalhem nos mesmos dados ao mesmo tempo, Figura 6.

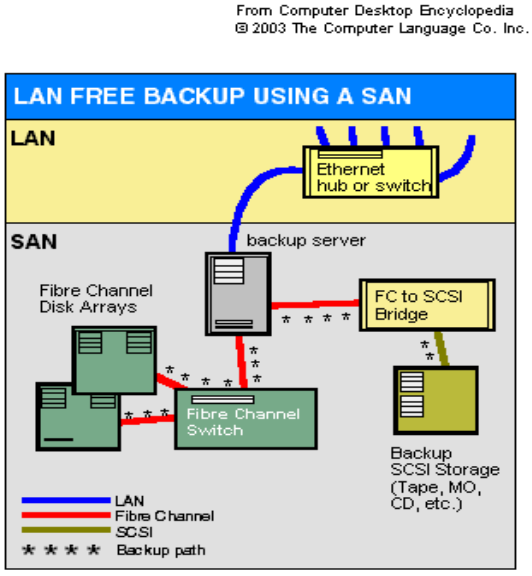


Figura 6. LAN-*free Backup* usando uma SAN, [10].

3.1.3. *Server-less Backup*

Server-less Backup é uma solução para SANs que foi projetada para reduzir custos de *hardware* e melhorar a eficiência, escalabilidade e tolerância a falhas. Esta arquitetura oferece vantagens distintas. Não sobrecarrega a rede corporativa, e faz o *backup* em SAN com rapidez. Geralmente, uma SAN configurada corretamente pode suportar a carga adicional de *backup* centralizado sem fazer grandes investimentos em equipamentos adicionais. Isso supõe, é claro, que o *hardware* da SAN é inteligente o suficiente para lidar com Backups *Server-less*. Um *Server-less Backup* através de uma SAN usa a switches da SAN ou diretores para lidar com o *backup*. Com efeito, a SAN é o servidor. Permite *backup* de disco-para-fita ou *backup* de disco para disco, sem depender de recursos do servidor ou largura de banda de rede.

Nesta arquitetura, Figura 7, ao servidor de fita é delegado o papel de "coordenador de sistema", ao invés de mover dados. Em uma configuração de *backup* do *Server-less*, um dispositivo de cópia assume a tarefa de realmente mover os dados do armazenamento em disco para a biblioteca de fita.

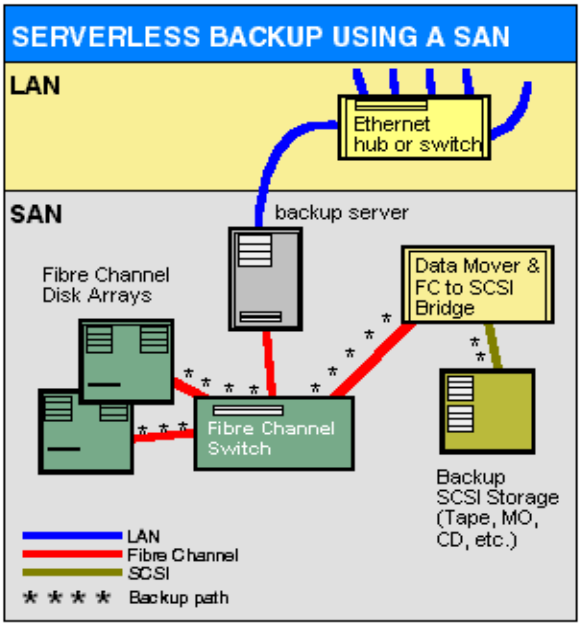


Figura 7. *Server-less Backup* usando uma SAN, [10].

Sintetizando, a arquitetura SAN/*Server-less Backup*, Figura 8, utiliza a tecnologia de clone normalmente disponível no *storage*. Faz-se um clone da base de produção e o servidor de *backup* faz o *backup* do clone independente da LUN de produção. Não há impacto na LUN de produção, mas a janela de *backup* pode ser grande neste caso.

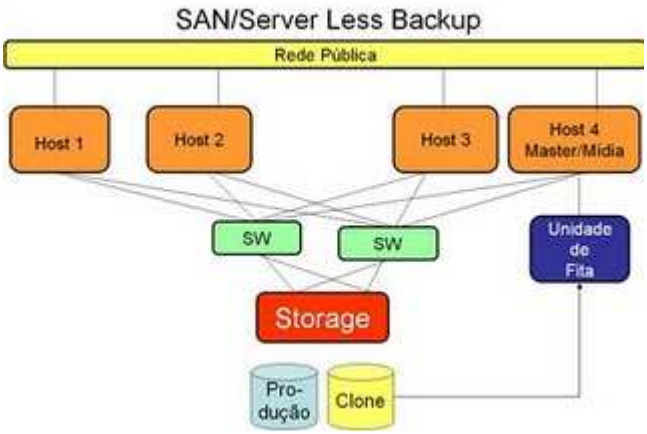


Figura 8. Arquitetura SAN/*Server-less Backup*, [10].

Server-less usa uma SAN, mas ao invés de um servidor de *backup* para gerenciar a operação, utiliza um "data mover", que é um simples dispositivo que pode ser incorporado na *storage* de *backup*. O servidor de *backup* inicia o *backup*, e o "data mover" gerencia.

3.2. Recursos de *Backup*

3.2.1. Servidor Principal

Um servidor para ponto de administração, Catálogo e Controle do Agendamento. Mantem um metadados do *backup*.

3.2.2. Servidores de Mídia

Vários Servidores. Conectam os dispositivos de *backup*. Trazem os dados do Servidor Principal e enviam para o dispositivo de *backup*. Podem fazer o *backup* de si mesmo. Meta dados viajam pela LAN para o Servidor Principal via Servidores de Mídia. O Mídia Server de SAN pode fazer *backup* de si mesmo.

3.2.3. Clientes do *Backup*

São os sistemas que sofrem o *backup*. Enviam os dados para o servidor de mídia.

3.2.4. Agentes

Fazem *hot-backup* (base de dados permance *on-line* para leitura e gravação durante o *backup*).

3.2.5. Mídias de *backup*

A mídia de armazenamento de *backup* pode ser uma unidade lógica, como um disco rígido, um dispositivo de armazenamento separado, como um disco removível, ou uma biblioteca inteira de discos ou fitas organizados controlados por alterador robótico (HD, DVD, CD-ROM, pen-drive, *storage* e fita).

Os dispositivos para armazenar dados, CDs, DVDs ou mesmo HDs e fitas possuem uma vida útil. Depois de muito tempo armazenado e sem uso, mesmo que seja no lugar ideal, acabam estragando e corrompendo os dados.

3.2.6. Disco

Nos últimos anos, os *drives* de disco nunca seriam usados como um meio de *backup*. No entanto, os preços de armazenamento caíram a um ponto que, em alguns casos, usar *drives* de disco para armazenamento de *backup* faz sentido. A razão principal para usar *drives* de disco como um meio de *backup* é a velocidade. Não há um meio de armazenamento em massa mais rápido. A velocidade pode ser um fator crítico quando a janela de *backup* do centro de dados é pequena e a quantidade de dados a serem copiados é grande.

3.2.7. Fita

A utilização de uma unidade de *backup* baseada em tecnologia de fita, hoje, LTO (D2T) para o *backup* e um *software* de gerenciamento faz parte hoje de quase toda solução de *backup*/*restore*. A fita foi o primeiro meio de armazenamento de dados removível amplamente utilizado. Tem os benefícios de custo baixo e uma capacidade razoavelmente boa de armazenamento. Entretanto, a fita tem algumas desvantagens. Ela está sujeita ao desgaste natural, sendo necessário manter o registro do uso das fitas para subtituí-las ao atingirem o fim de suas vidas úteis. E o acesso aos seus dados é sequencial, tornando a procura por um arquivo específico uma tarefa longa. Por outro lado, a fita é uma das mídias de armazenamento em massa mais baratas e carrega uma longa reputação de confiabilidade.

As unidades de fita são uma opção interessante apenas para quem precisa armazenar uma grande quantidade de dados, pois o custo por megabyte das fitas é bem mais baixo que o dos HDs e outras mídias. Segundo [11], é indicado que as fitas magnéticas são superiores aos discos rígidos em matéria de custo por volume de dados. O problema é que o custo do equipamento é relativamente alto e as fitas possuem tempo de validade, o que acaba obrigando o operador a fazer sempre pelo menos duas cópias para ter um nível maior de segurança. Em geral, é indicada para médias e grandes empresas. Nestas, as unidades de fita são utilizadas como parte de uma biblioteca de fitas comandada por um robô. Nas implementações baseadas em SAN quase sempre as unidades de fita são bibliotecas que podem ter vários *drivers* e utilizar diversos composições de cartuchos.

A tecnologia LTO é um formato de fita aberto desenvolvido pela HP, IBM e Seagate, que busca suprir a demanda crescente pela proteção de dados do mercado de servidores. A solução de armazenamento de dados no formato LTO Ultrium apresenta alta performance para *backup*, restaurações e aplicações de arquivamento. O formato LTO Ultrium possui um roteiro

definido para crescimento e escabilidade para cinco gerações - cada nova geração apresentará capacidade e velocidade dobradas, e possibilita integridade dos dados durante a evolução da fita.

Mais recentemente, a idéia de utilizar o *backup* em disco (D2D) vem ganhado adeptos, permitindo combinar o que há de melhor em ambas as tecnologias (D2D2T).

A opção hoje é a de utilizar uma combinação de discos e fita para *backup*. Os discos utilizados para o *storage* de *backup* são normalmente do tipo SATA para reduzir o custo da solução.

Um estudo realizado pelo Clipper Group, [12], compara os custos do *backup* realizado em fita LTO-3 com o *backup* em disco SATA. O estudo conclui que o *backup* em disco tem um custo de aquisição 6.5 vezes maior do que o do *backup* em fita. Além disso o espaço físico ocupado pelo *backup* utilizando discos no *storage* é 6.2 vezes maior do que o espaço ocupado pela unidade de fita e que o custo com energia e refrigeração é 25 vezes maior quando se utiliza *backup* em disco.

3.3. Proteção do *Backup*

É preciso ficar atento as orientações abaixo descritas para ser bem sucedido.

- Só pessoas autorizadas devem ter acesso ao local a fim de evitar roubos ou destruição dos *backups*.
- local deve ser protegido contra agentes nocivos naturais (poeira, calor, umidade). O ideal é que o local seja também à prova de fogo. No mercado existe cofres projetados para dar a máxima proteção às mídias de armazenamento, noentanto o custo é alto.
- Em libraries com leitor a laser as fitas devem ser rotuladas com uma etiqueta que apresente em código de barras a identificação do volume (8 a 6 caracteres alfanuméricos. Em geral, se usa as iniciais da empresa e um número sequencial - quatro ou três dígitos - seguidos da identificação da mídia, por exemplo, L3, para LTO Ultrium 3, Figura 9.

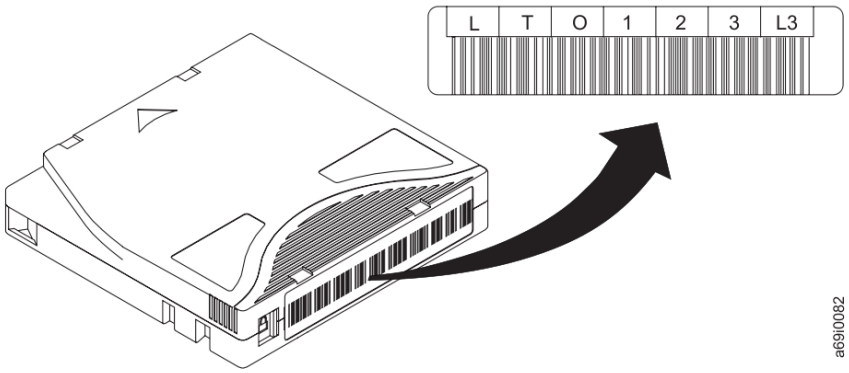


Figura 9. Código de barras para identificação da mídia LTO Ultrium 3, [25].

- No arquivamento das fitas, identifique-as com uma etiqueta onde deve constar informações como o número da fita, o nome do servidor, a data do *backup* e área "backpeada".

- Os *backups* devem ser verificados logo após a sua geração e, posteriormente, em intervalos regulares.

- Fazer cópia do *backup*. É melhor prevenir pois mídias de armazenamento podem se deteriorar com o tempo.

- Guardar cópias do *backup* fora do local de origem, de preferência, bem distante e seguro.

- Quando for necessário restaurar um sistema, recomenda-se, fazê-lo numa área diferente da original para não correr o risco de sobrescrever os dados mais recentes.

Instalações de segurança (*backup*) externas (*off-site*). A proteção de dados fora do local, *off-site*, é a estratégia de envio de dados críticos para fora do local principal (fora do site principal), como parte de um plano de recuperação de desastres. Os dados são normalmente transportados para fora do local utilizando meios de armazenamento removíveis, como fitas magnéticas, por exemplo, ou podem ser enviados eletronicamente através de um serviço de *backup* remoto. Envio de *backups off-site* garante que os sistemas e servidores possam ser recarregados com os últimos dados, no caso de uma catástrofe natural, erro acidental ou falha do sistema.

No caso de sinistro que danifique as instalações principais, enviabilizando o funcionamento operacional, deve-se garantir que as instalações de *backup* externas não sejam atingidas e opere em condições idênticas em, no máximo, 48 (quarenta e oito) horas após ter sido decretado o estado de contingência.

Embora algumas organizações gerenciem e armazenem seus próprios *backups off-site*, muitas optam por ter seus *backups* gerenciados e salvos por empresas que se especializam na proteção comercial. Uma preocupação adicional com os serviços de *backup* de terceiros é que

o custo depende da quantidade de dados de *backup*.

Os sistemas de *backup* remoto que funcionam através da Internet, normalmente através da criação de uma rede privada virtual (VPN), fazem *backup* apenas dos itens que foram alterados desde o *backup* anterior (incremental). Nestes é preciso levar em conta a largura de banda, o tempo e custo de lidar com as cópias de segurança para mensurar os custos. A maioria dos sistemas só fazer *backup* de arquivos ou pastas designados.

Capítulo 4
Escolha de Ferramenta para *Backup* Corporativo

Ao planejar um ambiente de *backup*, deve-se verificar se a configuração de *hardware* existente ou prevista é compatível com o *software* de *backup* a ser adquirido. A compatibilidade de *hardware* não é realmente garantida. É uma realidade sob medida que o *hardware* pode estar otimizado para o *software* proprietário, e vice-versa. Empresas consolidadas já apresentam seus produtos com base neste preceito. Mas a tarefa de implantar um *software* livre está muitas vezes relacionado a uma questões de experimentação e adaptação. Tem-se de verificar a compatibilidade com os recursos de *hardware*, versões requeridas de *software*, e testar a funcionalidade para executar as funções cotidianas de *backup* em um ambiente corporativo.

Dependendo da complexidade de um ambiente, pode não ser possível encontrar um produto que ofereça tudo que é necessário. Por menor que seja a organização, existe alguns compromissos em selecionar um produto de *backup*, e alternativas podem ser encontradas para alcançar o que é preciso de imediato, sem afastar um determinado produto, simplesmente porque não executa nativamente uma função específica que é necessária.

A proteção de sistemas não deve ser comprometida pela implantação de um sistema cujo os componentes em uso não foram devidamente testados, ou pela implantação de combinações de equipamentos que não foram certificados pelos fornecedores. Sem testes de verificação de funcionalidade do *software* ou dos equipamentos envolvidos a segurança fica comprometida e pode resultar em problemas ainda piores dos que já existem, criando uma situação desagradável, posteriormente.

Ao avaliar o potencial das novas ferramentas de *backup*, é necessário avaliar a funcionalidade básica oferecida pelos produtos e determinar quais as ofertas mais adequadas ao ambiente. Analisar os custos adicionais pelas funcionalidades não nativas. Montar uma lista de verificação exaustiva dos requisitos de funcionalidade. Estes devem, então, serem submetidos a diferentes fornecedores de *backup* para que possam explicar o que cada produto pode ou não fazer. Uma alternativa funcional pode não ser necessariamente uma solução ideal para um determinado recurso, mas no geral, considerando-se soluções alternativas, é possível obter um sistema talvez até superior às necessidades.

4.1. Comparação entre Softwares de *Backup*

Numa comparação de ferramentas de *backup* existirão sempre discrepâncias entre as várias funcionalidades oferecidas e as requisitadas. Devem ficar claros quais são os recursos

requisitados e quais são os prioritariamente desejados para iniciar uma comparação. Invariavelmente a escolha de um produto de *backup* deve ser uma decisão baseada no compromisso de maior proteção.

Como regra a considerar na avaliação de uma nova ferramenta de *backup* e que jamais se deve assumir qualquer responsabilidade sobre um produto de *backup* que ainda está sob avaliação. O certo é comprar uma ferramenta de *backup* depois de um período de análise razoavelmente compreensivo. Desta forma, verificando se uma funcionalidade desejada ou requerida realmente corresponde as espectativas da empresa.

Há ainda algumas considerações de outra ordem, mas que devem ser levadas em conta para o sucesso da implantação de um novo sistema de *backup*: treinamento, suporte técnico e de manutenção, documentação.

Tabela 1. Comparação de características entre softwares de *backup*.

	Bacula	**Amanda**	**Tivoli (TSM)**	**BrightStor ARCserve**
Níveis de *Backups*	Completo, Diferencial, Incremental, Consolidação	Completo, Diferencial Incremental	Completo, Incremental	Completo, Diferencial, Incremental
Formato dos Dados	Próprio (Aberto)	Aberto, recuperável sem o Amanda	Próprio (Fechado)	Próprio (Fechado)
Suporte aos Autochangers	Sim	Sim	Sim	Sim
***Backup* para Fitas e Discos rígidos**	Sim	Sim	Sim	Sim
***Backup* para DVDs**	Sim	Sim	Não	Não
Staging	Sim	Sim	Sim	Sim
Catalogo em SQL	Sim	Não	Sim	Não
Open Source	Sim	Sim	Não	Não
Suporte Comercial	Sim	Sim	Sim	Sim
GUI	Sim	Sim	Sim	Sim
Multiplatforma	Sim	Sim	Sim	Sim
Antivírus	Não	Não	Não	Sim
***Backup* Expande-se por Múltiplos Volumes**	Sim	Sim	Sim	Sim
Relatórios	Sim	Sim	Sim	Sim
Notificações	Sim	Sim	Sim	Sim
Encriptação do Fluxo de Dados	Sim (TLS)	Sim	Sim	Sim
Suporte a Agentes de *Backup* para MySQL/Oracle	Não	Sim	Sim (produto adquirido à parte)	Sim (produto adquirido à parte)
Catálogo	MySQL, SQLite PostgreSQL	MySQL	DB2	Ingres

De acordo com a Tabela 1, adaptada de [13], pacotes de *software* de *backup open source* podem ser capazes de atender às funcionalidades básicas requisitadas pelo ambiente corporativo, minimizando os custos e dando maior controle sobre suas funcionalidades. Acima de tudo, segundo [14], o principal é restringir os requisitos ao que é realmente necessário, e comprovar se estes pacotes podem satisfazer essas necessidades através de estudos de viabilidade. *Software open source* permite liberdade de visualização, alteração e distribuição do código fonte.

4.2. *Software* Proprietário versus *Open source*

O *software* é protegido pelas leis de direito autorais, que garantem ao titular o direito de uso e exploração econômica exclusiva sobre o programa. O *software* livre se fundamenta no direito autoral com a diferença de que o autor opta por permitir aos usuários usar, copiar, modificar e redistribuir o programa por ele criado seja gratuitamente ou com custo, disponibilizando o código fonte. Enquanto que no *software* proprietário o modelo tradicional impõe diversas restrições ao usuário quanto à sua utilização, modificação e acesso ao código fonte. A licença do *software* proprietário geralmente é concedida mediante o pagamento de royalties.

É importante não confundir *software* livre com *software* grátis porque a liberdade associada ao *software* livre de copiar, modificar e redistribuir, independe de gratuidade. Existem programas que podem ser obtidos gratuitamente mas que não podem ser modificados, nem redistribuídos, como pode ser visto na Figura 10.

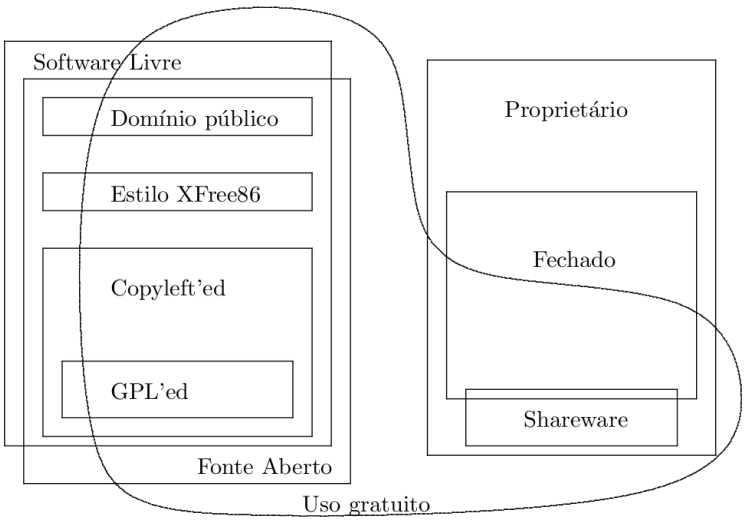


Figura 10. Categorias de licenças de *software*, [15].

Além disso, o *software* livre também exige uma licença de uso específica para assegurar a liberdade do usuário. O idealizador do projeto GNU, Richard Stallman, criou a

Free Software Foundation (Fundação do *Software* Livre) para tratar dos aspectos jurídicos e organizacionais do projeto.

Para o movimento *open source*, o fato do programa de computador ser de "fonte aberta" é uma questão de cunho prático e, para o *software* livre é de cunho ético, compreendendo uma mobilização sócio-política. As licenças de *open source* são variáveis quanto à utilização do uso do código-fonte, enquanto as licenças de *software* livre possuem uma forte característica: a necessidade de compatibilidade com a GNU GPL ou com as chamadas quatro liberdades: executar, estudar, redistribuir e aperfeiçoar o programa, como pode ser visto no Anexo A.

As licenças dos programas de computador são criadas para suprimir a liberdade de compartilhar ou modificar o *software* já que ele é considerado um produto de venda e seus direitos autorais são protegidos por lei. Para maiores detalhes quanto aos tipos de licenças existentes, consultar o Anexo B.

No Brasil, é a Lei número 9609/98 que dispõe sobre a proteção da propriedade intelectual de programa de computador e sua comercialização no país.

A Lei diz, ainda, que o uso de programa de computador é objeto de contrato de licença. Por sua vez, a violação dos direitos de autor de programa de computador pode resultar em detenção e/ou multa.

A partir da década de 1980, alguns setores da indústria de informática iniciaram oposição ao modelo de *software* proprietário, defendendo a liberdade de acesso ao código fonte a fim de possibilitar o compartilhamento da informação e o desenvolvimento tecnológico, introduzindo no mercado o conceito de *software* livre.

Ao final da década de 1990, alguns países como Índia, Japão, China e Coréia do Sul passaram a adotar políticas de favorecimento à utilização do *software* livre, como forma de incentivo ao desenvolvimento tecnológico e democratização do acesso a novas tecnologias.

O uso de *Software* Livre em governos tem crescido rapidamente em todo o mundo.

O Governo Federal vem adotando uma política de favorecimento e incentivo ao uso e desenvolvimento do *software* livre no Brasil. O Projeto de Lei nº 2.269/99, em seu artigo 1º, estabelece que as empresas estatais e de economia mista, as empresas públicas e todos os demais organismos públicos ou privados sob controle da sociedade brasileira estão obrigados a utilizar *preferencialmente software* livre em seus sistemas e equipamentos de informática. No mesmo sentido, determina o artigo 9º que apenas será permitida a utilização de *software* proprietário na ausência de *software* livre que atenda aos requisitos estabelecidos na licitação pública.

O governo federal tem interesse em *software* livre por desobrigar-se da dependência de plataformas fechadas que impõem uma dependência econômica, com renovações de licenças anuais de *software*, por ter maior controle do processamento das informações, pois baseiam-se no princípio da transparência, permitindo auditoria plena e por estimular a modificação de softwares que sejam cada vez mais sob medida.

Segundo [16], “o acesso ao código, permite à Administração maior liberdade, tanto na escolha de fornecedores de serviços, quanto nas opções de customização de programas, abrindo leque de oportunidades de acesso aos contratos administrativos. O governo federal já percebeu que *software* livre é um bom negócio. O dinheiro fica no país, gerando emprego e renda e não é remetido para o exterior pela compra de licenças”.

A Universidade Estadual de São Paulo (USP), a Universidade de Brasília (UnB), a Universidade Federal do Rio de Janeiro (UFRJ) e o Tribunal de Contas da União (TCU) são exemplos de aplicação do *software* livre na esfera pública.

Segundo [17], Recife foi a primeira cidade do mundo a contar com uma legislação que regulamenta o uso preferencial de softwares de código aberto por secretarias e órgãos públicos. A lei, de autoria do secretário de Desenvolvimento Econômico, Waldemar Borges, foi sancionada pelo prefeito João Paulo. "Além de economia, a lei representa para a Prefeitura maior soberania e independência tecnológica", conforme Waldemar Borges, [17]. "Nossa intenção é socializar o conhecimento e colocá-lo à disposição da população", declarou o prefeito João Paulo.

Capítulo 5
Bacula

Bacula é um *software* de *backup open source* projetado para ser escalável, abrangendo desde um único computador a redes corporativas. Permite ao administrador do sistema realizar o gerenciamento centralizado de *backups*, a recuperação e verificação de dados em uma rede de computadores heterogênea.

O Bacula é ideal para quem busca uma solução de *backup* em rede, flexibilidade, serviços de catálogo, além de recursos adicionais próprios a ambientes corporativos. Requer do administrador conhecimentos avançados em administração de SOs Unix e Linux e/ou experiência com outras ferramentas sofisticadas de *backup*. Apesar disso, segundo o manual, [5], um grande número de usuários relata que o Bacula é simples de configurar e usar diante de programas equivalentes.

Sua arquitetura modular fornece uma solução de *backup* robusta e completa para uma variedade de sistemas Unix e Linux, Windows e MAC OS X. O Bacula também pode ser executado inteiramente em um único computador e pode fazer *backup* de vários tipos de mídias, incluindo fitas e discos.

Segundo [18], o Bacula foi originalmente escrito por John Walker e Kern Sibbald em 2000. John deixou o projeto pouco tempo depois do seu começo, e Kern, atualmente o principal mantenedor, trabalhou sozinho neste projeto de meados do ano 2000 até o primeiro lançamento do Bacula ao público em Abril de 2002. Desde então outros desenvolvedores, distribuídos em várias localidades do mundo, têm contribuído com trabalho adicional.

De acordo com informações do projeto publicado no Source Forge, [3], Figura 11, desde Abril de 2002, o Bacula teve mais de 1 milhão de downloads, o que é quatro vezes mais do que qualquer outro programa de *backup* de código aberto durante o mesmo período. De acordo com as estatísticas de download, isso o torna como programa de *backup open source* mais popular.

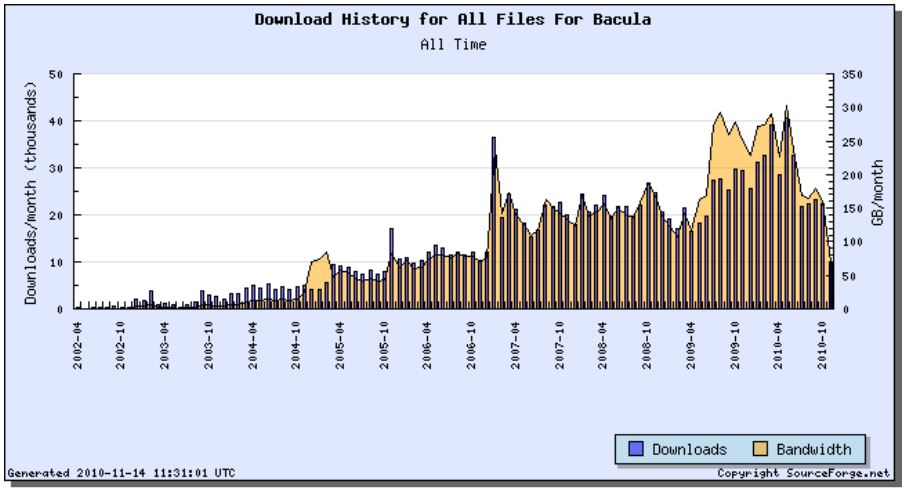


Figura 11. Histórico de downloads do Bacula, [3].

Segundo o manual, [5], a grande maioria do código fonte do Bacula, desenvolvida em C/C++, está licensiada sob os termos da GNU Public License versão 2. Também são encontradas outras licenças, que se relacionam a utilização do manual, códigos fontes de bibliotecas de terceiros, utilitários de domínio público e copyright.

5.1. Componentes do Bacula ou Serviços

A comunicação entre os módulos do Bacula baseia-se no modelo cliente-servidor, tendo o protocolo TCP/IP como elemento de interconexão em rede. Esta filosofia do projeto Bacula, permite que os componentes sejam implantados separadamente em várias máquinas, com o objetivo de considerar o planejamento de capacidade, os níveis de acesso e a distribuição dos hardwares de uma empresa. Como pode ser visto na Figura 12, o Bacula é composto dos seguintes cinco grandes componentes ou serviços: o Director, o Console, o File Daemon, o Storage Daemon, e os serviços de monitoramento.

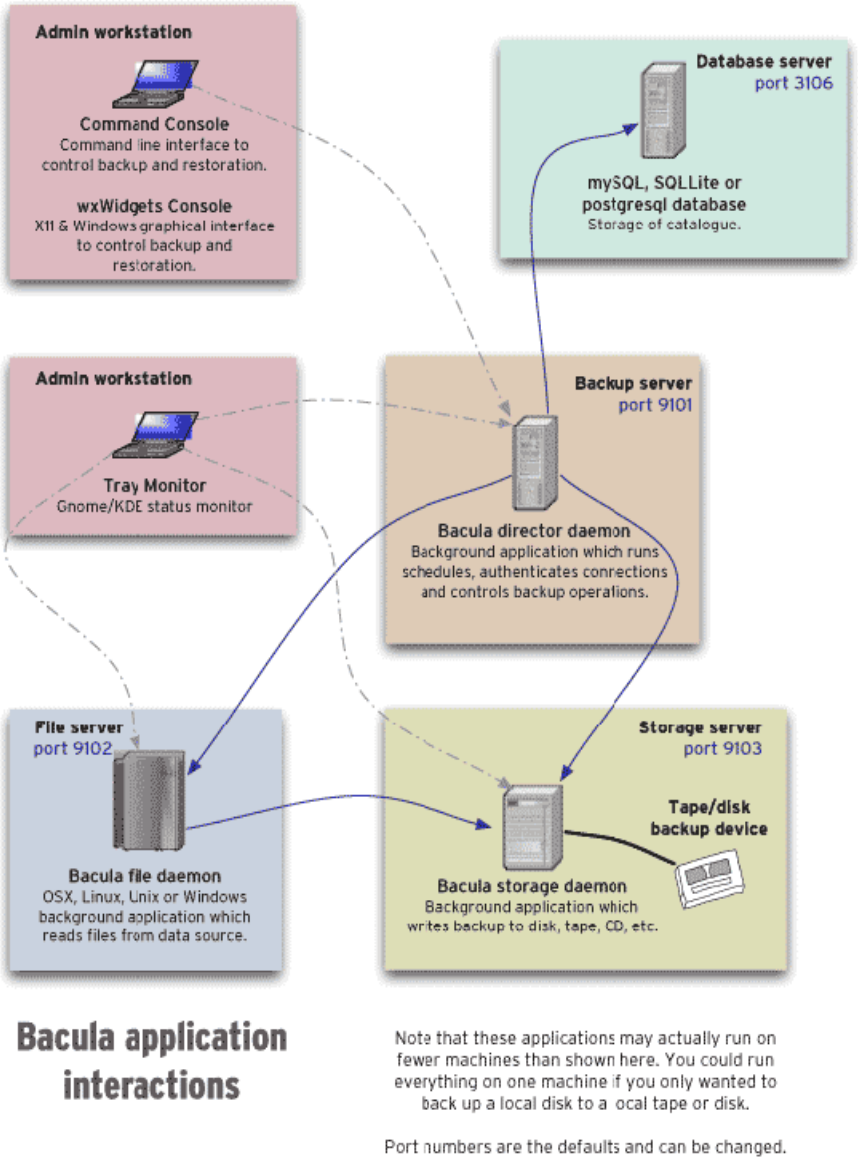


Figura 12. Estrutura do Bacula e relação entre os respectivos componentes, [5].

Os componentes do *software* Bacula, ilustrados pela Figura 12, cumprem funções específicas básicas dentro do ambiente. Abaixo, cada componente é descrito com vistas à aplicação global.

- **Director:** É o módulo central do Bacula responsável por supervisionar e gerenciar os demais módulos, através de um arquivo de configuração que reune informações pertinentes à política de *backup*, como por exemplo, a definição de *jobs*, pools, agendamentos, controles de acesso. É executado como um *daemon* (ou serviço) em segundo plano.

- **File Daemon:** É o módulo que é instalado em cada máquina na qual se deseja que os *backups* de seus arquivos sejam feitos, assumindo desta forma, o papel de cliente. Através de uma ordem vinda do Director, este módulo passa a enviar os arquivos e respectivos atributos diretamente aos dispositivos de armazenamento. É executado em segundo plano como um *daemon* em diversos SOs para os quais foi portabilizado.

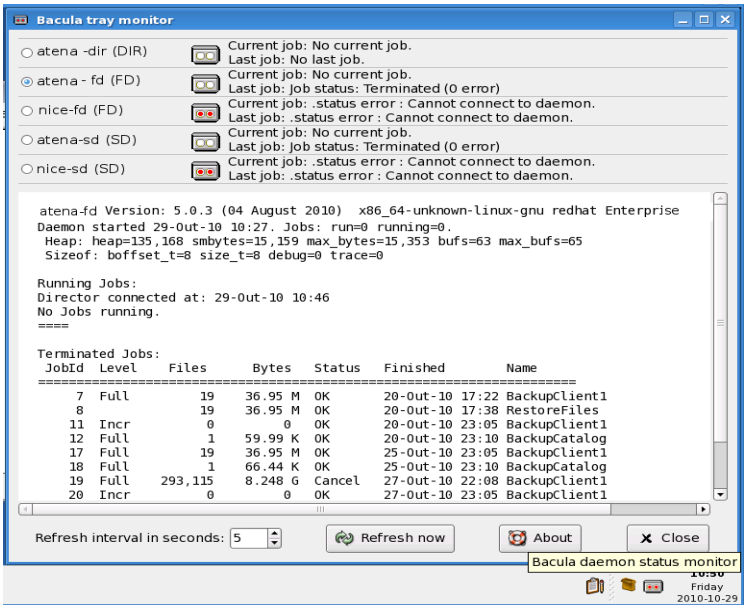
- **Storage Daemon:** A função deste módulo é a de controlar os dispositivos de armazenamento. Este controle é exercido através da suas interações via *scripts* com utilitários que interfaceiam comandos do *hardware* com o SO, como o *mt* e o *mtx*. É responsável, por exemplo, pela definição de *autochangers*, de *drives* de gravação de fitas, de rótulos, pelo carregamento e descarregamento de mídias, ou na ausência de fitas, na definição e utilização de um diretório do sistema de arquivos no qual serão armazenados os *backups*. Também é executado como um *daemon*.

- **Console:** Este componente é utilizado para operações manuais e alguns ajustes do servidor e demais módulos, em tempo de execução. Com o console administrativo, tornam-se possíveis, por exemplo, operações de gererenciamento de *jobs*, manipulação de mensagens e informações de *status*, bem como, o gerenciamento de volumes. Está disponível em todas as plataformase é a interface padrão da maioria dos administradores para a interação com o Bacula. Apresenta-se, basicamente, como um console em modo texto, ainda que, haja muitos projetos que o utilizam sob interfaces gráficas para *desktops* e servidores web, como poderá ser visto na seção 5.2.

- **Catalog:** Este componente é responsável por catalogar os arquivos que foram armazenados pelo *backup*, de modo que as informações de localização dos arquivos, dos *jobs* e volumes envolvidos, sejam mantidos em bancos de dados relacionais. Os serviços deste módulo tornam possíveis a rápida localização e restauração de qualquer arquivo desejado. Os três SGBDs atualmente suportados são: MySQL, PostgreSQL e SQLite. Estes SGBDs estão disponíveis para vários sistemas operacionais e oferecem um número considerável de recursos, incluindo a indexação rápida, consultas arbitrárias e segurança. Esta compatibilidade com diferentes SGBDs decorre de que o Bacula utiliza o framework libdbi no código fonte que implementa a interação com o catálogo.

- **Monitor:** Este componente permite ao administrador visualizar graficamente os sinais de *status* emitidos pelo(s) Director(s), File Daemo(s) e Storage Daemon(s). Necessita da API GTK para suas janelas serem exibidas e manipuladas nos *desktops* gráficos GNOME, KDE ou outros que suportem o padrão FreeDesktop.org de bandeja do sistema. É também chamado de tray-monitor, nome dado ao executável, devido a sua localização na bandeja do sistema, como pode ser visto na Figura 13.

a.



b.

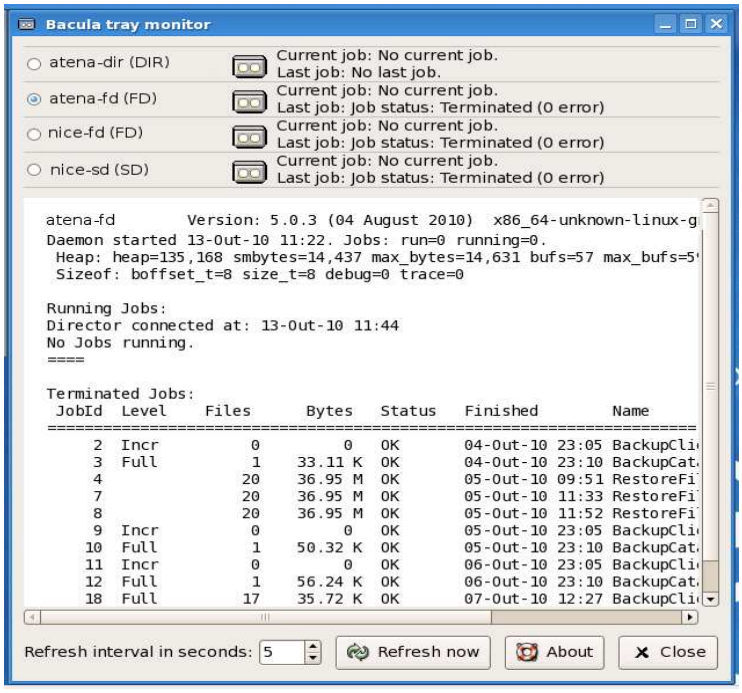


Figura 13. O tray-monitor exibindo sinais de *status* dos módulos do Bacula: a. quando há erros; b. quando não há erros.

5.2. Outras Interfaces Gráficas

O Bacula, atualmente, dispõe de interfaces gráficas que utilizam as funcionalidades oferecidas pelo console administrativo, a exemplo do Bat e do bwx-console, baseados no QT

e no wxWidgets, respectivamente.

Estas interfaces gráficas são acessíveis apenas pelos *desktops* gráficos GNOME e KDE. Desta forma, para permitir a administração remota destas interfaces gráficas, faz-se necessário que um serviço de VNC esteja instalado e com acesso remoto assegurado aos modos gráficos dos servidores.

Nenhuma das interfaces gráficas, utilizadas neste trabalho, podem ser consideradas completas. Todas se apresentam em desenvolvimento e têm como parâmetro de evolução o console em modo texto.

O Bat, mostrado na Figura 14, conserva muitas das funcionalidades do console administrativo.

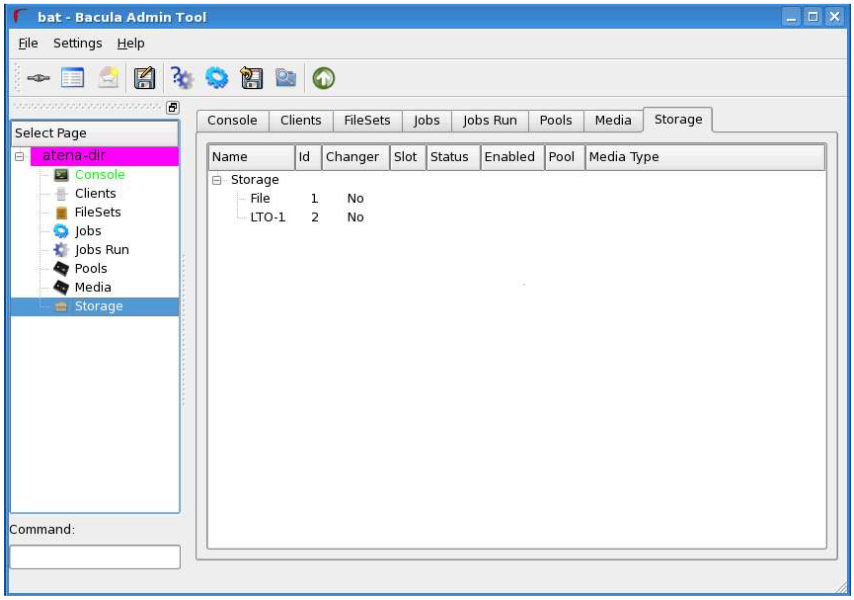


Figura 14. O Bat.

O bwx-console, Figura 15, é bem mais simples que a interface gráfica anterior, pois é desprovida de recursos de configuração interativa de *jobs*, volumes, *pools*, clientes e *storages*. Conta apenas com uma janela simples para passagem de comandos ao console em modo texto, e com um meio interativo para selecionar arquivos para *restores* serem realizados. Possui o recurso *tab completion* e fornece ajuda instantânea ao comando que está sendo digitado.

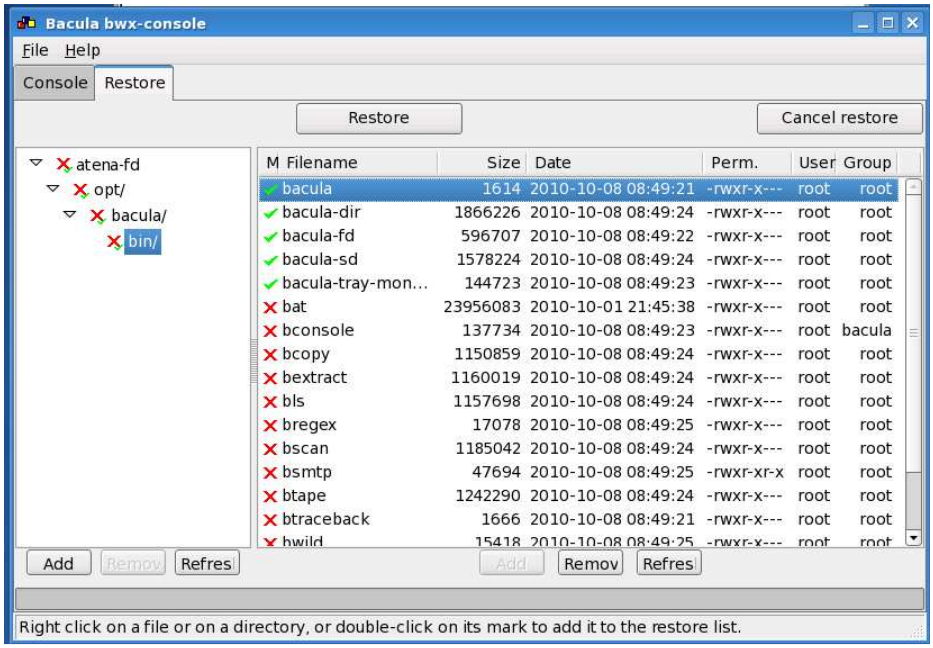


Figura 15. O bwx-console.

Como exemplo de interface gráfica acessível por WEB Browsers, encontra-se o Bweb, Figura 16. Esta interface gráfica, distribuída com o nome de pacote bacula-gui-5.0.3.tar.gz, é o projeto oficial do Bacula para promover a interação entre o console e o servidor WEB apache, o que o torna menos dependente do acesso físico às máquinas para uma administração remota do sistema de *backup*/*restore*. Nota-se que, em termos de recursos, o Bweb é um pouco melhor que o Bat. Há ainda a possibilidade de extrair estatísticas dos *jobs* realizados.

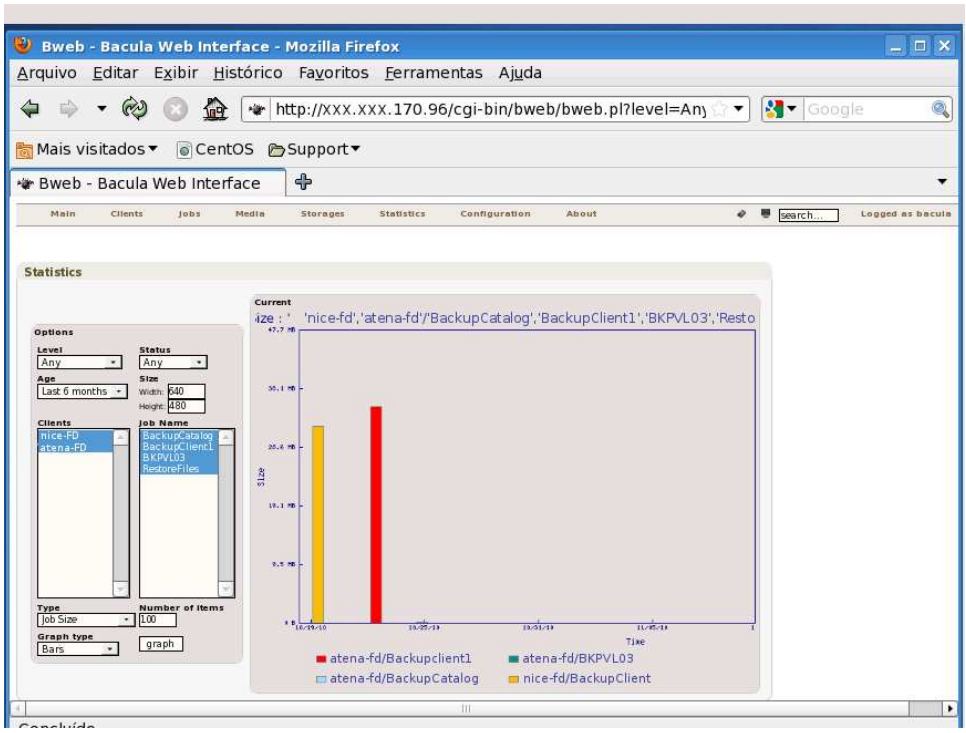


Figura 16. O Bweb.

5.3. Configuração do Bacula

Os arquivos de configuração do Bacula abrangem a definição de toda a arquitetura do sistema de *backup*/*restore*. Cada arquivo de configuração está associado a um módulo e descreve a disposição destes, os recursos de *hardware* consumidos, e partes da política de *backup*, Figura 17.

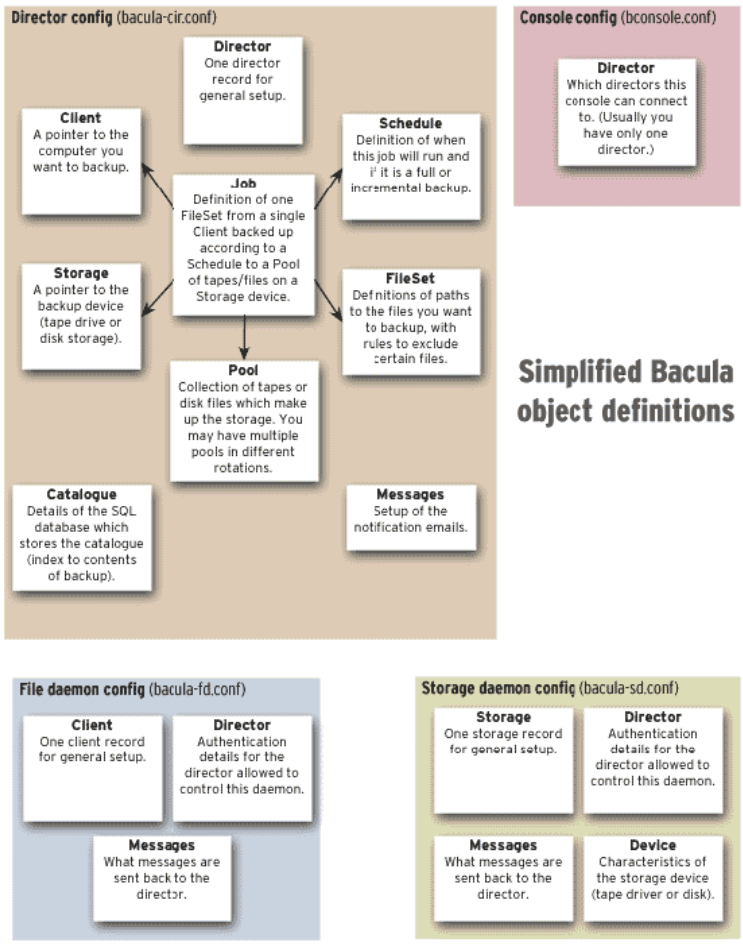


Figura 17. Definição simplificada dos objetos Bacula, [5].

Numa sintaxe bem próxima a C/C++, os arquivos de configuração consistem em um ou mais conjuntos de instruções de definição de recursos, diretivas, entre chaves “{}”, que descrevem objetos Bacula e como o sistema deve lidar com esses objetos.

Cada componente ou módulo do Bacula tem um arquivo de configuração individual no diretório onde foi instalado, .../bacula/etc, podendo ser editado com qualquer editor de texto padrão, embora a configuração padrão fornecida com o pacote represente um modelo configuração de base.

Os componentes do Bacula, Figura 18, interagem entre si por meio de senhas de autorização definidas em cada módulo pela diretiva password, além de outras diretivas, como Name, Device, MediaType. Por exemplo. a senha do recurso Storage no arquivo */etc/bacula/bacula-dir.conf* deve coincidir com a senha do recurso Director no arquivo */etc/bacula/bacula-sd.conf*.

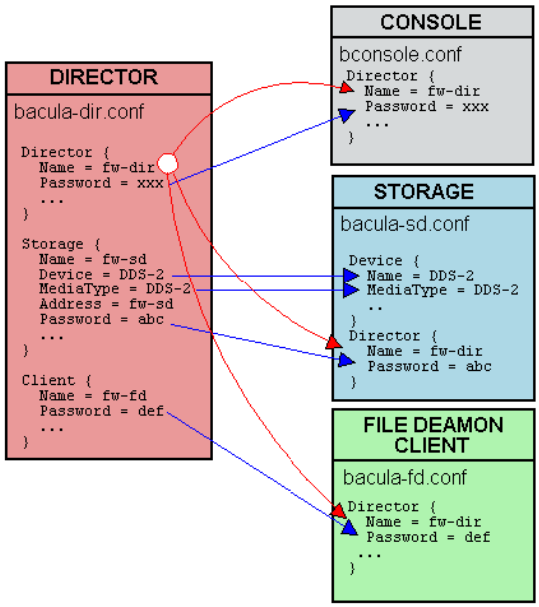


Figura 18. Relação de diretivas entre arquivos de configuração, [5].

Cada um dos módulos do Bacula comunica-se uns com os outros através de um protocolo baseado em TCP/IP, cuja autenticação padrão é baseada em senhas em texto puro, geradas aleatóriamente na instalação do Bacula.

O Bacula inclui também suporte ao TLS, garantindo que o tráfego entre o File Daemon, o Director, e o Storage Daemon não seja transmitida em texto puro. Além disso, o TLS pode fornecer outra camada de autenticação no topo da exigência de senhas correspondentes nos daemons.

A configuração do Bacula, pode tomar inicialmente um certo tempo, a depender da complexidade do projeto do sistema de *backup*/*restore*. Mas, uma vez que as fases de configuração, testes e implantação forem completadas, só irão ser necessárias novas alterações dos arquivos de configuração quando os requisitos do projeto do sistema e da política de *backup* também forem modificados, como por exemplo, acréscimo de clientes ou de filesets.

Numa implantação comum do Bacula, há somente um Director. O arquivo *bacula-dir.conf*, o qual controla o Director, tem como foco a definição dos *jobs*. Por sua vez, a definição dos *jobs* de *backup* depende de quais são os clientes, quais são os filesets, em qual pool, onde os *backups* serão armazenados e quando serão executados.

O Storage Daemon manipula todas as comunicações com os dispositivos da arquitetura de *backup*. Assim como o Director, comumente há apenas um único Storage Daemon sendo executado a partir de uma instalação Bacula, mas este Daemon pode ter muitos dispositivos de *backup* definidos em seu arquivo de configuração, *bacula-sd.conf*. Pode ser instalado em cada máquina que possui interfaces de comunicação com um dispositivo de armazenamento específico, como as tape libraries, ou em máquinas cujo meio

de armazenamento são os próprios discos rígidos, por exemplo. Por padrão, a configuração original do Bacula contempla a realização de *backups* e *restores* D2D no mesmo *host*.

O File Daemon é instalado em cada máquina cliente. Seu arquivo de configurações é chamado *bacula-fd.conf*. Este módulo promove um meio de comunicações entre o cliente e o Storage Daemon, transportando todos os arquivos, os quais possui acesso, para compor os *backups* que devem ser realizados.

Apesar de uma grande instalação, exigir a edição significativa dos arquivos de configuração, os testes de Bacula realizados no capítulo seguinte utilizam os mesmos princípios que seriam aplicados em maior escala.

5.4. Interações entre os Serviços Bacula

Durante a execução, o Director instrui o File Daemon para se comunicar diretamente com o Storage Daemon adequado, removendo assim o diretor do caminho de alto fluxo de dados de *backup*. O diretor mantém o seu próprio serviço de agendamento para que os trabalhos possam prosseguir (em horários fixos), sem intervenções que partam de um console ou de comandos do usuário. As saída das tarefas agendadas são enviadas via e-mail para um ou mais administradores definidos na configuração do diretor.

Segundo o diagrama de blocos do manual, [5], Figura 19, as interações típicas entre os serviços Bacula para um trabalho de *backup* são mostradas. Cada bloco representa um processo separado em geral (normalmente um *daemon*). Em geral, o Director supervisiona o fluxo de informações. Ele também mantém o catálogo.

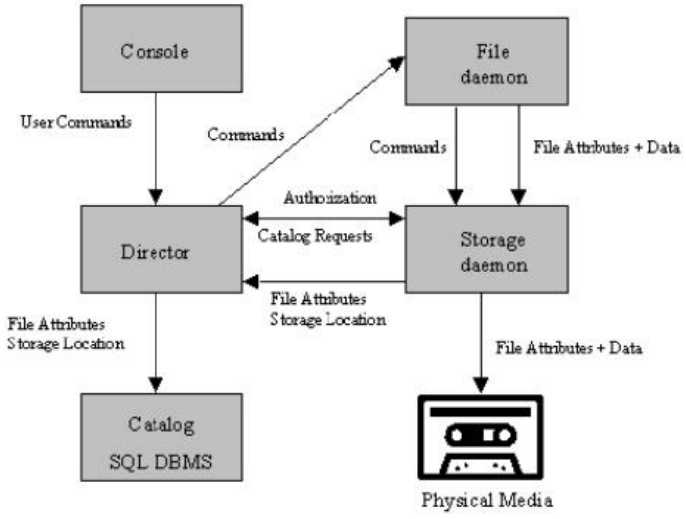


Figura 19. O fluxo de informações entre os componentes do Bacula em um trabalho de *backup* típico, [5].

5.5. Requisitos de Sistema

De acordo com o manual oficial, [5], e com a devida adaptação aos ambientes em teste, o Bacula possui os seguintes requisitos de sistema:

- GNU C++ versão 2.95 ou superior para a compilação. O Bacula tem sido compilado com sucesso com a versão 4.1.3 do GNU C++. Pelo fato de que o GNU C++ é um pacote rpm separado do GNU C, ambos precisam estar instalados. Em sistemas baseados no RedHat, como por exemplo o CentOS, o compilador C++ faz parte do pacote gcc-c++, em rpm.

- Pacotes de terceiros podem ser necessários para o Bacula. Com exceção do MySQL e do PostgreSQL, estes pacotes podem ser encontrados agrupados nos pacotes depkgs e depkgs-qt, disponíveis na seção de downloads no site oficial, [4]. No entanto, na maioria das versões atuais dos sistemas Linux e FreeBSD, estes pacotes são fornecidos pelos repositórios nativos.

- As versões mínimas para cada SGBD suportado pelo Bacula são:

  - MySQL, versão 4.1

  - PostgreSQL, versão 7.4

  - SQLite 3

- Um sistema com suporte a pthreads.

- Compatibilidade com o padrão POSIX, no qual o código fonte foi projetado com o objetivo de favorecer a portabilidade.

- O programa bwx-console depende do GTK+-2.x e do wxWidgets. Este programa também foi desenvolvido e testado com as últimas versões estáveis dos padrões ANSI e Unicode.

- O programa tray-monitor foi desenvolvido para o GTK+-2.x. Este programa requer que o desktop GNOME seja menor ou igual à versão 2.2, enquanto que o desktop KDE, maior ou igual a versão 3.1, ou qualquer outro gerenciador de janelas que suporte o padrão FreeDesktop para programas que residem em bandejas de sistema.

- O Bacula tem sido compilado e executado nos sistemas OpenSuSE Linux, FreeBSD e Solaris. Porém, o Bacula é totalmente compatível com qualquer GNU/Linux, o que significa, por exemplo, distribuições Linux como RedHat, CentOS, Fedora, Debian, Madriva, Gentoo, Ubuntu, em arquiteturas 32/64 *bits*, entre outros.

Capítulo 6
Estudo de Caso

Neste capítulo, o Bacula será abordado sob uma perspectiva prática com base em uma experiência realizada em um ambiente real de uma empresa de médio porte. O objetivo é demonstrar a aplicabilidade do Bacula em ambientes corporativos, e a experiência na sua implantação. As etapas de instalação e configuração serão relatadas, bem como exemplos de testes realizados que evidenciaram mudanças de comportamento dos *backups* e *restores* por meio de alterações realizadas nos arquivos de configuração.

Inicialmente, serão descritos dois ambientes distintos, onde a funcionalidade do Bacula foi verificada, e os requisitos de *software* necessários à etapa de instalação do mesmo.

6.1. Análise do Ambiente

Segundo [4], a lista de compatibilidade com os *autochangers* foi criada a partir da contribuição dos usuários que tiveram contato com hardwares corporativos e foram bem sucedidos com a utilização do Bacula nestes ambientes.

Para o propósito deste trabalho, pudemos verificar a funcionalidade do Bacula em um ambiente corporativo real, em dois ambientes de *hardware* distintos:

- Ambiente 1: Dell PowerEdge 2600, [19], SUN StorageTek L20, [20], SO com arquitetura de 32 *bits*.
- Ambiente 2: IBM BladeServer HS21 8853, [21], IBM Library TS3200 L4U, [22], SO com arquitetura de 64 *bits*.

Não há relatos de testes de compatibilidade que envolva os hardwares descritos nestes dois ambientes. Portanto, a demonstração da compatibilidade destes dois ambientes representa uma novidade à comunidade de usuários do Bacula.

A seguir encontra-se uma descrição maior a cerca dos hardwares delimitados anteriormente em dois ambientes.

6.1.1. Ambiente 1

Os componentes de *hardware* deste primeiro ambiente caracterizam-se por estarem no mercado corporativo a mais de cinco anos. Porém, a utilização de servidores com características semelhantes ainda é comum nas empresas, pois nem todas estão dispostas a arcar com o alto custo requerido com a aquisição de equipamentos modernos e equipe técnica preparada para enfrentar um processo de migração para uma nova infraestrutura.

Na empresa na qual foi realizado o estudo, a biblioteca de fitas StorageTek L20 da Sun, Figura 20.a, possui um *drive* que lê e grava fitas com tecnologia LTO Ultrium de primeira geração. Esta biblioteca de fitas foi adquirida com limitações de *hardware* e *firmware*, e, de um total de 20 *slots* de fitas, apenas 10 estão habilitados para o trabalho.

a. b.

Figura 20. a. Sun StorageTek L20; b. Dell PowerEdge 2600.

As informações armazenadas pelas fitas LTO-1 gravadas com a L20, datam desde 2004. Devido à importância destas informações, as fitas necessitarão, em breve, passar por um processo de migração do seu conteúdo para serem adequadas ao ambiente 2, descrito na próxima subseção.

Através de um cabo SCSI, o *drive* da biblioteca de fitas liga-se localmente à interface Ultra 160 SCSI de um servidor DELL PowerEdge 2600, Figura 20.b. Esta ligação permite que a velocidade alcançada na transferência dos dados ao *drive*, alcance até 160 MB/s. Uma vez que o *drive* foi alcançado, os dados seriam gravados em fita uma velocidade constante de no máximo 20 MB/s, segundo a Tabela 2, [23].

Tabela 2. Principais atributos das quatro primeiras gerações de tecnologia LTO.

	Geração			
Atributo	LTO-1	LTO-2	LTO-3	LTO-4
Data de Lançamento	2000	2003	2005	2007
Capacidade Nativa de Dados	100 GB	200 GB	400 GB	800 GB
Velocidade Máxima (MB/s)	20	40	80	120
Suporte a compresão	Sim, “2:1”			

Contudo, para a empresa, este ambiente não encontra perspectiva futura de continuidade uma vez que não há condições técnicas para manter uma conexão física via interface SCSI com a nova infraestrutura sem realizar um upgrade dos componentes internos tanto da L20 quanto do servidor PE2600 para se adequarem a tecnologia FC.

Anteriormente à chegada da nova infraestrutura, este ambiente era responsável pelo gerenciamento de *backup*/*restore* através do aplicativo BrighStor ARCserve v.9 da Computer

Associates, instalado na plataforma Linux, Red Hat 8.0. Além dos clientes que se comunicavam via Ethernet, protocolo TCP/IP como os servidores de sistemas de desenvolvimento e produção.

Podemos conferir na Tabela 3, as características do servidor PE2600, cujo nome do *host* é nice. Para efeitos de teste com o Bacula, foi utilizado o sistema operacional CentOS, versão 5.5 32 *bits*, num esquema de dual-boot com o sistema anterior, RedHat 8.0.

Tabela 3. Características do servidor PE2600.

Nome do *Host*: nice
Características de *Hardware*
Processor: 2 x Intel XEON CPU 2.40GHz @ 2.39GHz (Total Cores: 4), Motherboard: Dell PowerEdge 2600, Chipset: Intel E7500 Hub, Memory: 4 x 512 MB DDR-200MHz, Disk: 18GB LD 0 RAID1 17G + 182GB LD 1 RAID5
Características de *Software*
OS: CentOS release 5.5 (Final), Kernel: 2.6.18-194.el5 (i686), Desktop: KDE

Ao invés de utilizar o RedHat Enterprise Linux, recomendado pelo suporte oficial do Bacula, foi utilizado o CentOS, um *rebuild* livre do primeiro feito pela comunidade de *software* livre. Esta escolha se justifica, pois, para fins de teste, foi adotado um sistema operacional bastante semelhante, sem custo com a aquisição de licença de subscrição e amplo acesso aos repositórios gratuitos.

6.1.2. Ambiente 2

Este segundo ambiente caracteriza-se por apresentar componentes de *hardware* modernos, utilizados atualmente em médias e grandes empresas.

A IBM BladeServer HS21 8853, também chamada por *blade* como um designativo geral à categoria, é um servidor destinado à computação de alto desempenho em formato de lâmina com um design modular otimizado para minimizar o uso do espaço físico e energia, como pode ser visto na Figura 21.a.

a. b.

Figura 21. a. IBM *blade* HS21; b. BladeCenter H Series.

A *blade* utilizada encontra-se na baia 2 de um módulo BladeCenter H Series, Figura 21.b, e está conectada a 2 *switchs* ethernet e a 1 *switch* FC de 4Gbps. Estes 2 *switchs* ethernet correspondem as redes X.X.170.0/24 e X.X.6.0/24. O *switch* FC realiza a ligação das blades, com as tape-libraries e o *storage* por meio do estabelecimento de zonas de interfaces.

Em termos de velocidade de transferência de dados interdispositivos, o FC de 4Gbps utilizado nesta infraestrutura atinge a taxa de 800 MB/s, segundo [24].

Na Tabela 4, podemos ver detalhadamente algumas características desta unidade de processamento. O sistema operacional utilizado foi o CentOS, versão 5.5, para a arquitetura de 64 *bits*.

Tabela 4. Características da *blade* 2.

Nome do *Host*: atena
Características de *Hardware*
Processor: 2 x Intel Xeon CPU E5420 @ 2.50GHz (Total Cores: 8), Motherboard: IBM eServer BladeCenter HS21 -[8853PTM]-, Chipset: Intel 5000P Chipset Hub, Memory: 4 x 4096 MB DDR, Disk: 73GB
Características de *Software*
OS: CentOS release 5.5 (Final), Kernel: 2.6.18-194.17.1.el5 (x86_64), Desktop: KDE

Há 2 IBM libraries TS3200 3573 L4U, Figura 22.a. Cada uma suporta 48 *slots* de capacidade divididos em 4 magazines, agrupamentos de 12 *slots*, Figura 22.b.

a. b.

Figura 22. a. Duas IBM Libraries TS3200 L4U; b. Magazines esquerdas e direitas da TS3200.

Cada uma destas tapes libraries possuem 2 *drives* Full-High com tecnologia LTO Ultrium de quarta geração para trabalhar com fitas LTO Ultrium 2, no modo apenas leitura, e com fitas LTO Ultrium 3 e 4, nos modos leitura e gravaçao, como pode ser visto na Tabela 5, [25]. De um total de 48 *slots*, são opcionalmente reservados apenas 3 *slots* dedicados à I/O Station e 1 *slot* para o cartucho de limpeza.

Tabela 5. Compatibilidade entre as tecnologias de *drives* e fitas.

Drive de fitas IBM Ultrium	Fitas IBM TotalStorage LTO Ultrium			
	800 GB (Ultrium 4)	400 GB (ultrium 3)	200 GB (Ultrium 2)	100 GB (Ultrium 1)
Ultrium 4	Leitura/Escrita	Leitura/Escrita	Leitura	–
Ultrium 3	–	Leitura/Escrita	Leitura/Escrita	Leitura
Ultrium 2	–	–	Leitura/Escrita	Leitura/Escrita
Ultrium 1	–	–	–	Leitura/Escrita

Este ambiente pertence a uma SAN. O dipositivo de armazenamento em rede é o IBM System Storage DS4700 Express. Chamado comumente por *storage*, esta solução de *hardware*, destina-se a fornecer grande capacidade de armazenamento em discos, compartilhados em rede, através de poderosos recursos de gerenciamento.

No *storage* há 4 *enclosures* (gavetas) Exp810, Figura 23: a primeira unidade contém 16 discos FC, a segunda 4 discos SATA e 12 discos FC, a terceira 16 discos FC e a quarta 16 discos SATA.

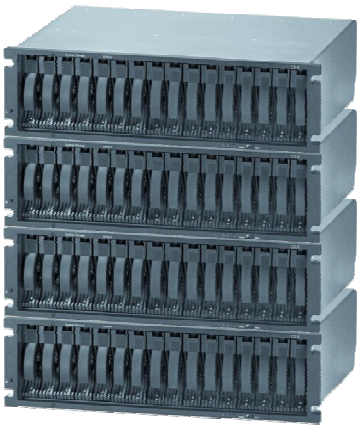


Figura 23: Quatro *enclosures* Exp810 compondo o IBM System Storage DS4700.

Estes discos, também chamados volumes físicos, juntos fornecem uma capacidade total, em escala de Terabytes. São agrupados em arrays que são divididos em volumes lógicos, cada um recebendo um LUN para o devido mapeamento.

6.1.3. Arquitetura de Rede

Os dispositivos de *hardware* dos ambientes 1 e 2 interagem entre si e com outras máquinas através da topologia de rede, exposta na Figura 24.

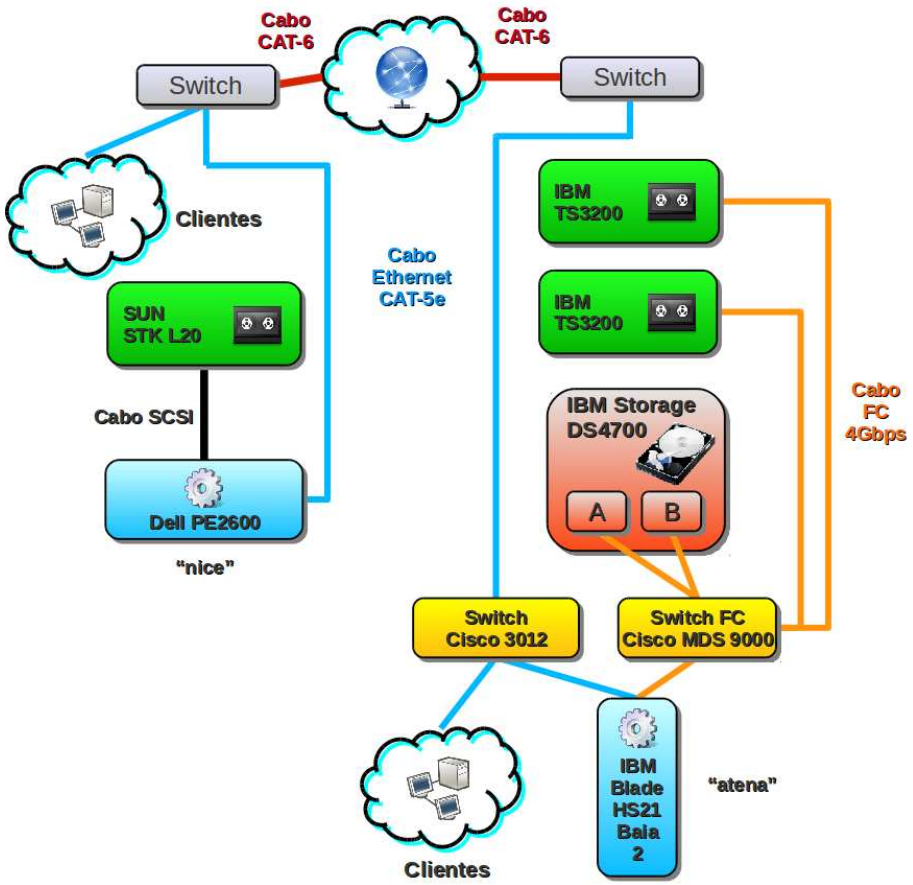


Figura 24. Arquitetura de rede que envolve os ambientes 1 e 2.

Pode ser percebido, através da figura acima, que a arquitetura de rede da empresa é majoritariamente do tipo LAN *Backup*, o que não é aconselhável para a maioria das empresas de médio porte.

A arquitetura LAN-*free Backup* fica restrita ao *backup* de dados cujos volumes lógicos do *storage* estejam mapeados diretamente para o servidor de *backup*, localizado na *blade 2*. A SAN fica restrita a conexão que o *switch* FC fornece que são: cada *blade* comunica-se individualmente com as controladoras A e B do *storage*, mais a comunicação entre a *blade* 2 e as bibliotecas TS3200.

Através de testes realizados com o *software* Iperf, a largura de banda entre o *host* nice e o *host* atena atinge cerca de 95,4 Mbps, ou seja, 11,93 MB/s. Esta velocidade ocorre apenas quando ambas se comunicam através da rede 170. Isto ocorre porque a interface de rede eth1 do *host* nice, configurada para a rede 170, refere-se à uma placa Intel Ethernet Pro 100, de largura de banda de 100 Mbps.

Comunicando o *host* nice com o *host* atena através da rede, a largura de banda chega à 936 Mbps, ou seja, 117 Mbps. Desta vez o *host* nice estava atrelado a rede 6, através da interface eth0, que refere-se a uma placa Intel Gigabit Ethernet.

Portanto, para testes de *backup* em rede, a interface eth0 deve receber prioridade na transferência dos dados. Caso contrário a biblioteca de fitas poderá sofrer problemas com a diminuição da velocidade de gravação dos dados por meio dos *drives*.

O procedimento de análise das melhores rotas para a realização de *backups* e *restores* garante que os dados sejam manipulados pelos *drives* sem ocorrer perda de desempenho da velocidade. Pode-se comprovar na prática, que a taxa de velocidade de uma determinada rota da rede é determinada pelo dispositivo de menor desempenho.

6.2. Requisitos de Sistema

Através de varias tentativas realizadas para se obter uma instalação estável, bem como, abranger o maior número de recursos possíveis, foram adicionados ao SO CentOS 5.5 os seguintes grupos de pacotes necessários para compilar o código fonte do Bacula 5.0.3, como pode ser visto resumidamente na Figura 25.

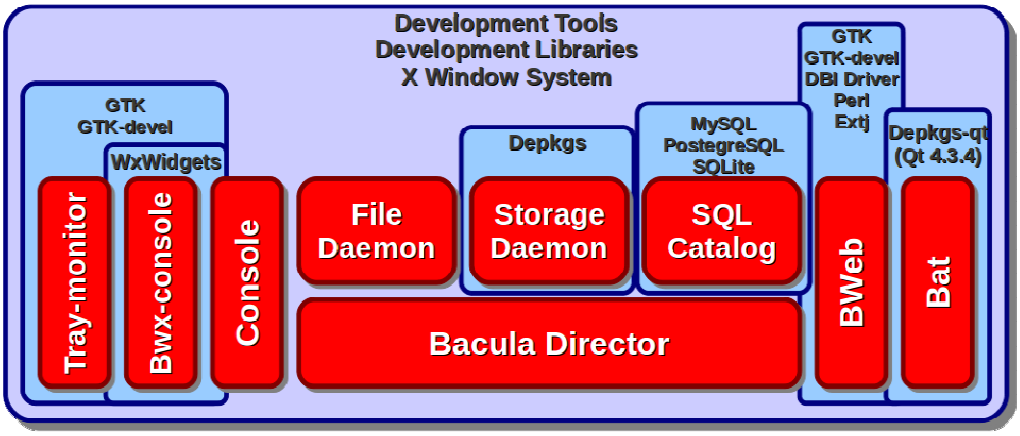


Figura 25. Diagrama de dependência de pacotes por módulos do Bacula.

De acordo com a figura acima, as principais dependências de pacotes possuem a coloração azul, enquanto que os módulos do Bacula estão em vermelho.

6.3. Instalação

Para os propósitos de teste, utilizamos o CentOS, versão 5.5, como sistema operacional para a instalação do código fonte do Bacula, na sua mais recente versão 5.0.3. Pacotes rpm também poderiam ser utilizados, porém, impossibilitaria determinadas configurações personalizadas, além de que as versões em pacotes binários não acompanham o mesmo ritmo de lançamento das versões em código-fonte e nem sempre estão disponíveis para todas as arquiteturas e distribuições Linux.

Depois de baixar e descompactar o código fonte do Bacula, é recomendado baixar e descompactar também os pacotes de dependências, depkgs e depkgs-qt. Estes dois últimos quando compilados fornecem alguns requisitos de *software* essenciais a uma compilação bem-

sucedida do Bacula, como pode ser visto na Tabela 6, abaixo.

Tabela 6. Requisitos de *software* essenciais ao Bacula.

Pacotes de terceiros	depkgs	depkgs-qt
mt	X	
mtx	X	
tapeinfo	X	
qt		X
qwt		X

Após instalar as dependências, é preciso executar o *script* de configuração da compilação do Bacula. Este *script* de configuração é bem documentado e produz uma saída de debug detalhada. É possível passar para o *script* de configuração uma grande quantidade de atributos que personalizam a instalação, assinalando quais componentes devem estar presentes para o funcionamento do sistema. Exemplos de configurações podem ser obtidos diretamente do site oficial, contudo algumas pequenas alterações foram feitas tanto para organizar a distribuição de pastas de instalação tanto adicionando determinadas funcionalidades, como pode ser visto no trecho abaixo.

```
[...]
  Database type:                  MySQL
[...]

  Bacula conio support:           yes -lncurses
  readline support:               no
  TCP Wrappers support:           no
  TLS support:                    yes
  Encryption support:             yes
  ZLIB support:                   yes
  enable-smartalloc:              yes
  enable-lockmgr:                 no
  bat support:                    yes
  enable-gnome:                   no
  enable-bwx-console:             yes wxWidgets 2.8
  enable-tray-monitor:            yes
  client-only:                    no
  build-dird:                     yes
  build-stored:                   yes
  Plugin support:                 yes
  AFS support:                    no
```

```
  ACL support:                    yes
  XATTR support:                  yes
  Python support:                 no
  Batch insert enabled:           yes

[...]
```

Como se pode observar, esta saída retrata os recursos escolhidos na etapa de configuração da instalação de um servidor completo de *backup* Bacula. Com esta configuração serão instalados, todos os módulos: File Daemon, Storage Daemon, Director e Console. Além dos suportes a compressão e a encriptação de dados e de rede, são instaladas três interfaces gráficas nativas do projeto Bacula: o Bwx-console, o Tray-monitor e o Bat.

Dentre as três opções possíveis de SGBDs, foi escolhido o MySQL, por questões de popularidade entre os tutoriais sobre o assunto encontrados na internet, o que não representa uma escolha ideal ou otimizada.

Em termos de análise de benchmark, foge ao escopo deste trabalho a discussão sobre qual SGBD é o mais eficiente para ser utilizado junto ao Bacula. Alguns sites, por exemplo, [26], sugerem que em média, o desempenho da união Bacula e PostgreSQL seja melhor que com o MySQL. Em outros, a discussão se mantém equilibrada, onde a escolha depende da aplicação e do tipo de ambiente utilizados, [27], [28]. Já o BaculaSystems, [29], transparece um posicionamento pró-MySQL.

Salvo as diferenças entre as arquiteturas de *hardware* e SO, em 32 *bits* e 64 *bits*, respectivamente para os ambientes 1 e 2, a configuração do Bacula é bastante semelhante em ambos, uma vez que serão instalados servidores completos, cada um possuindo pelo menos os quatro módulos principais.

Uma vez que o *script* de configuração é executado, o Bacula é compilado e instalado como a maioria dos softwares *open source*, usando o procedimento normal *make; make install*. Depois de compilado, os executáveis do Bacula podem ser encontrados na pasta *bin* do diretório de instalação. A execução do *script* do Bacula inicia os módulos Storage Daemon, File Daemon e Director.

Em seguida, são realizados alguns passos de pós-instalação a fim de que sejam geradas a base de dados e as tabelas, bem como, sejam ajustados alguns permissionamentos e senhas.

6.4. Testando o *drive* de fitas

Saber objetivamente se uma determinada biblioteca de fitas e seus *drives* funcionarão bem com o *software* de *backup* escolhido é questão de encontrar uma lista de compatibilidade entre *software* e *hardware* que responda a esta dúvida.

O manual oficial do Bacula, [5], conta com a divulgação de usuários que realizaram

testes de compatibilidade e que obtiveram sucesso no uso do Bacula com hardwares de classe corporativa. Como se pode observar no site, [4], a lista de compatibilidade entre o Bacula e as bibliotecas de fitas é bastante pequena, o que motiva àqueles que dispõem de ambientes semelhantes para experimentarem uma solução *open source* de *backup*.

Antes de prosseguir com o teste, é necessário saber se pelo menos as bibliotecas de fitas e os *drives* foram reconhecidos pelo SO como pode ser notado nos trechos abaixo:

•No *host* nice:

```
[root@nice ~]# cat /proc/scsi/scsi
Attached devices:
[...]
Host: scsi1 Channel: 00 Id: 00 Lun: 00
  Vendor: STK      Model: L20              Rev: 0214
  Type:   Medium Changer                   ANSI SCSI revision: 03
Host: scsi1 Channel: 00 Id: 15 Lun: 00
  Vendor: SEAGATE  Model: ULTRIUM06242-XXX Rev: 1619
  Type:   Sequential-Access                ANSI SCSI revision: 03
[root@nice ~]# dmesg |grep scsi
[...]
scsi 1:0:0:0: Attached scsi generic sg4 type 8
scsi 1:0:15:0: Attached scsi generic sg5 type 1
st 1:0:15:0: Attached scsi tape st0
[root@nice ~]#
```

•No *host* atena:

```
[root@atena ~]# cat /proc/scsi/scsi
Attached devices:
[...]
Host: scsi1 Channel: 00 Id: 02 Lun: 00
  Vendor: IBM      Model: ULT3580-TD4      Rev: A239
  Type:   Sequential-Access                ANSI SCSI revision: 03
Host: scsi1 Channel: 00 Id: 02 Lun: 01
  Vendor: IBM      Model: 3573-TL          Rev: 9.20
  Type:   Medium Changer                   ANSI SCSI revision: 05
Host: scsi1 Channel: 00 Id: 03 Lun: 00
  Vendor: IBM      Model: ULT3580-TD4      Rev: A239
  Type:   Sequential-Access                ANSI SCSI revision: 03
Host: scsi1 Channel: 00 Id: 04 Lun: 00
  Vendor: IBM      Model: ULT3580-TD4      Rev: A239
  Type:   Sequential-Access                ANSI SCSI revision: 03
Host: scsi1 Channel: 00 Id: 05 Lun: 00
  Vendor: IBM      Model: ULT3580-TD4      Rev: A239
  Type:   Sequential-Access                ANSI SCSI revision: 03
Host: scsi1 Channel: 00 Id: 05 Lun: 01
```

```
  Vendor: IBM      Model: 3573-TL          Rev: 9.20
  Type:   Medium Changer                   ANSI SCSI revision: 05
[...]
[root@atena ~]# dmesg |grep scsi
scsi0 : qla2xxx
scsi1 : qla2xxx
[...]
scsi 1:0:2:0: Attached scsi generic sg2 type 1
scsi 1:0:2:1: Attached scsi generic sg3 type 8
scsi 1:0:3:0: Attached scsi generic sg4 type 1
scsi 1:0:4:0: Attached scsi generic sg5 type 1
scsi 1:0:5:0: Attached scsi generic sg6 type 1
scsi 1:0:5:1: Attached scsi generic sg7 type 8
[...]
st 1:0:2:0: Attached scsi tape st0
st 1:0:3:0: Attached scsi tape st1
st 1:0:4:0: Attached scsi tape st2
st 1:0:5:0: Attached scsi tape st3
[...]
[root@atena ~]#
```

Apesar do *host* atena ligar-se às bibliotecas de fitas por meio de cabos FC, segundo [30], comandos SCSI são predominantemente transportados pelo protocolo Fibre Channel Protocol (FCP) da camada de transporte da mencionada rede. Desta forma, bibliotecas de fitas e discos de armazenamento SCSI usufruem de altas velocidades e da possibilidade de conexão com um grande número de dispositivos.

Através dos trechos acima, pôde-se distinguir os *drives* dos robôs de fitas, denotados pelos tipos Sequential-Access e Medium Changer, respectivamente.

Os *drives* e o robô das bibliotecas de fitas são reconhecidos como dispositivos SCSI e, desta forma, poderão ser encontrados sob a pasta */dev* do SO CentOS, conservando a nomenclatura *sg#* (onde # representa um número). Em especial, os drives serão utilizados pela nomenclatura *st#* ou por *nst#*, por exemplo, como */dev/st0* ou */dev/nst0*, caso deseje-se utilizar o primeiro *drive*. A diferença entre estas nomenclaturas para os *drives* indica que ao final de qualquer operação realizada a fita seja rebobinada ou não.

As informações acima mencionadas são bastante úteis para o utilitário *btape* executar uma série completa de testes de compatibilidade. Este utilitário inclui operações de leitura, escrita e busca em fitas, além de usar o arquivo de configuração do Storage Daemon como parâmetro de entrada. Isto implica que uma configuração simples do Storage Daemon, deve estra disponível antes de algum teste ser realizado com o utilitário *btape*.

Desta forma, os dispositivos detectados foram adicionados como recursos no *bacula-sd.conf*. Esta adição de recursos seguiu os exemplos comentados dentro do próprio *bacula-sd.conf*. Um trecho do *bacula-sd.conf* com o acréscimo de recursos para a StorageTEK L20, pode ser visto no trecho abaixo. Procedimento similar ocorreu com a IBM TS3200, sendo

reservadas as devidas alterações para as diretivas que envolvem o caminho lógico para os dispositivos SCSI e o tipo de mídia utilizada pelos *drives*.

```
Autochanger {

  Name = STKL20-Autochanger
  Device = STKL20-Drive-0
  Changer Command = "/opt/bacula/scripts/mtx-changer %c %o %S %a %d"
  Changer Device = /dev/sg4
}

Device {
  Name = STKL20-Drive-0
  Media Type = LTO-1
  Archive Device = /dev/nst0
  AutomaticMount = yes
  AlwaysOpen = yes;
  RemovableMedia = yes;
  RandomAccess = no;
  AutoChanger = yes
}
```

Uma maneira de saber se os arquivos de configuração permanecem corretos, gramaticalmente e sintaticamente, é invocar os executáveis do Bacula passando o parâmetro "-t", mais os respectivos arquivos de configurações para serem analisados.

Logo depois do *bacula-sd.conf* ser modificado, o *btape* foi executado com comandos que puderam simular gravações e leituras de blocos de dados nas fitas. Como todos os testes foram completados com êxito pelo *btape*, pôde-se comprovar que ambas as bibliotecas de fitas são compatíveis com o Bacula e foram configuradas corretamente, cuja listagem pode ser conferida em mais detalhes no Apêndice A.

6.5. Configurações e Testes

Como exposto no Capítulo 5, o Bacula possui arquivos de configuração para cada um de seus módulos. Nesta seção, serão mostrados exemplos de configurações baseados em testes simples de *backup* e *restore*, no Bacula.

Os testes que se seguirão às configurações permitirão demonstrar a realização de *backups* e *restores* locais, D2T, o que pode ser visto com maiores detalhes na Tabela 7, abaixo.

Tabela 7. Testes que serão realizados.

Nº	Ação	Dispositivo de Destino	Director	File Daemon	Storage Daemon
1	*Backup*	Sun STK L20	nice	nice	nice
2	*Restore*				
3	*Backup*	IBM TS3200	atena	atena	atena
4	*Restore*				

Para outros tipos de testes que compartilhassem em rede, clientes e dispositivos entre todos os módulos, seria preciso utilizar esquemas que discriminassem o compartilhamento de senhas com o objetivo de orientar quais as alterações devem ser realizadas nos arquivos de configurações.

De acordo com a Figura 26, abaixo, as senhas compartilhadas mantém a conexão entre todos os módulos, em casos de testes mais abrangentes, com variação de clientes e *storages*. Portanto, para demais testes é necessário adicionar clientes e dispositivos de armazenamento nos arquivos de configuração do Bacula.

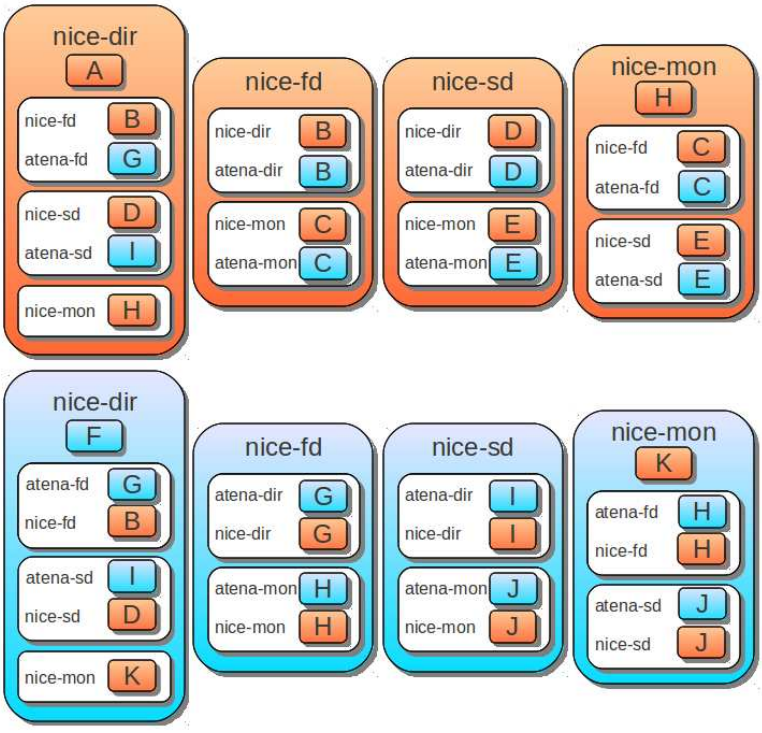


Figura 26. Esquema de compartilhamento de senhas.

Através da Figura 26, observa-se que as senhas são representas por uma única letra. As senhas que conservam a mesma coloração do módulo no qual se situam, são senhas nativas geradas no processo de instalação e, desta forma, pertencem ao *host* de mesma cor. As senhas que, em termos de cores, diferem do módulo no qual se situam, dizem respeito ao processo de

compartilhamento de senhas intermódulos. Não foram geradas novas senhas. Todas são sequencias aleatórias em forma de texto simples.

Fazendo-se as devidas correspondências entre as senhas, é percebido que o processo acaba por compartilhar os módulos File Daemon e Storage Daemon entre os Diretores. Desta forma, tanto o nice-dir quanto o atena-dir acessam o nice-fd através da senha "B", por exemplo.

Os procedimentos manuais realizados para todos os testes de *backup* propostos na Tabela 7 seguem, em comum, a seguinte metodologia:

1. O módulo Console é inicializado pelo executável *bconsole*. Um asterisco marca a posição para a inserção de comandos.
2. O comando *run* é executado.
3. O *bconsole* lista todos os *jobs* definidos para serem escolhidos.
4. O *job* desejado é selecionado e o Bacula mostrará uma pequena lista de configurações, como por exemplo, nome do cliente, tipo de *backup*, dispositivo de armazenamento e quando será realizado.
5. Se tudo estiver correto, digita-se *yes* e o *job* é realizado. Caso contrário, digita-se *mod* para configurar cada um dos atributos do *job*.
6. O módulos tem o *status* alterado e mensagens de *log* tornam-se disponíveis. O *job* pode ser rotulado como “Running” se está sendo executado, ou “Wating” acrescido de uma mensagem que informa que o *job* está aguardando sua vez na fila de *jobs*, por exemplo.
7. Depois da execução, o *job* pode ser rótulado com “OK” caso tenha sido bem sucedido. Pode assumir também os *status* “Canceled” ou “Error”, caso a execução tenha sido abortada via intervenção humana ou problemas de configuração, respectivamente.

Para os testes de *restore*, os procedimentos manuais conservam a maioria das etapas de *backup*, exceto em:

2. O comando *restore* é executado.
3. O *bconsole* lista várias opções para restaurar determinados arquivos, através do nome, de consultas SQL ou pelos números de identificação dos *jobs* de *backup*, JobId. Dependendo da opção selecionada, os arquivos desejados para serem restaurados podem necessitar serem marcados ou desmarcados, no caso dos indesejados.
4. As configurações gerais do *job* de restauração de arquivos são exibidas, como por exemplo, diretório de destino, cliente de destino.

Como exemplo dos procedimentos realizados, os testes 1 e 2 são descritos abaixo. A Figura 27 esquematiza os testes 1 e 2.

Figura 27. Esquema dos testes 1 e 2.

Na configuração do Director, há 3 *jobs* pré-definidos: o primeiro pode fazer um *backup* incremental da pasta dos binários do próprio Bacula, cujo nome é BackupClient1, o segundo pode fazer *backups* completos do catálogo, BackupCatalog, e o terceiro restaura arquivos dos *backups* realizados, RestoreFiles. Os dois primeiros *jobs* são automaticamente agendados, enquanto que o terceiro é invocado manualmente. Observe no trecho abaixo, que inicialmente os volumes não possuem nomes e são tachados como desconhecidos.

```
[root@nice bin]# ./bconsole -c ../etc/bconsole.conf
[...]
*status all
[...]
Scheduled Jobs:
Level          Type     Pri  Scheduled          Name               Volume
===================================================================================
Incremental    Backup    10  29-Nov-10 23:05    BackupClient1      nice-TMP
Full           Backup    11  29-Nov-10 23:10    BackupCatalog      nice-TMP
====
[...]
```

Um novo recurso Fileset foi criado, muito embora, o corrente *job* apresentasse um FileSet padrão, "Full Set", que continha 33,55 MB pertencentes a 18 arquivos da subpasta /bin do diretório de instalação do Bacula, bem como, com a opção padrão de cálculo de checagem de paridade MD5 ativa.

O novo Fileset, chamado Teste-Arquivo, contém um arquivo de 1,3 GB, que simula a utilização de um arquivo real. No trecho abaixo podemos observar a configuração padrão alterada.

```
FileSet {
  Name = "Teste-Arquivo"
```

```
  Include {
    Options {
      signature = MD5
    }

    File = /root/arquivo_teste.dmp.gz
  }
}
```

Contudo, o *job* BackupClient1 foi editado pelo console e executado para cada um dos testes de *backup* da Tabela 7. Como pode ser observado, foram editadas as diretivas “Level”, “Pool” e “Storage”, que indicam respectivamente o tipo de *backup*, qual o pool de mídias e qual o dispositivo de armazenamento. A diretiva FileSet indica quais serão os arquivos e pastas, cujo *backup* deverá ser feito.

Antes:

```
[...]
JobName:  BackupClient1
Level:    Incremental
Client:   nice-fd
FileSet:  Full_Set
Pool:     File (From Job resource)
Storage:  nice-File (From Job
resource)
When:     2010-11-29 18:48:35
Priority: 10
[...]
```

Depois:

```
[...]
JobName:  BackupClient1
Level:    Full
Client:   nice-fd
FileSet:  Teste-Arquivo
Pool:     Tape-LTO-1 (From User
input)
Storage:  nice-STKL20-LTO-1 (From
user selection)
When:     2010-11-29 18:48:35
Priority: 10
[...]
```

Em particular, mesmo que a diretiva “Level” seja “Incremental”, o primeiro *backup* deste *job* automaticamente elevaria o “Level” para “Full”, seguindo a lógica de que é necessário ter-se pelo menos um *backup* completo à frente de uma série de *backups* incrementais.

Logo depois que o *job* é confirmado, o Director espera que um nome de volume já tenha sido atribuído a um Pool, a fim de que o sistema de arquivos ou quaisquer outros dispositivos recebam permissão de acesso à escrita. Caso contrário o *job* é retido e o Director aguarda a nomeação do volume a ser utilizado pelo Storage Daemon.

Com a execução do *job*, os *status* dos módulos do Bacula envolvidos com o processo de *backup* são alterados, como por exemplo, o módulo nice-dir, Director do *host* nice, no trecho abaixo.

```
[...]
```

```
nice-dir Version: 5.0.3 (04 August 2010) i686-pc-linux-gnu redhat
Daemon started 29-Nov-10 18:46, 0 Jobs run since started.
 Heap: heap=135,168 smbytes=63,166 max_bytes=63,166 bufs=307 max_bufs=307
[...]
Running Jobs:
Console connected at 29-Nov-10 18:48
 JobId Level   Name                       Status
======================================================================
    18 Full    BackupClient1.2010-11-29_18.50.02_05 is running
====
[...]
```

Quando o *backup* recebe o *status* "OK", mensagens de *log* tornam-se disponíveis para a evolução do procedimento de *backup* e o comportamento de cada um dos módulos podem ser conferidos posteriormente.

```
[...]
*messages
[...]
29-Nov 18:53 nice-sd JobId 18: Job write elapsed time = 00:03:06, Transfer rate = 6.929
M Bytes/second
29-Nov 18:53 nice-dir JobId 18: Bacula nice-dir 5.0.3 (04Aug10): 29-Nov-2010 18:53:31

  Build OS:               i686-pc-linux-gnu redhat
  JobId:                  18
  Job:                    BackupClient1.2010-11-29_18.50.02_05
  Backup Level:           Full
  Client:                 "nice-fd" 5.0.3 (04Aug10) i686-pc-linux-gnu,redhat,
  FileSet:                "Teste-Arquivo" 2010-11-29 18:50:02
  Pool:                   "Tape-LTO-1" (From User input)
  Catalog:                "MyCatalog" (From Client resource)
  Storage:                "nice-STKL20-LTO-1" (From user selection)
  Scheduled time:         29-Nov-2010 18:48:35
  Start time:             29-Nov-2010 18:50:04
  End time:               29-Nov-2010 18:53:31
  Elapsed time:           3 mins 27 secs
  Priority:               10
  FD Files Written:       1
  SD Files Written:       1
  FD Bytes Written:       1,288,870,533 (1.288 GB)
  SD Bytes Written:       1,288,870,645 (1.288 GB)
  Rate:                   6226.4 KB/s
  Software Compression:   None
  VSS:                    no
  Encryption:             no
  Accurate:               no
```

```
Volume name(s):         LTO029
Volume Session Id:      1
Volume Session Time:    1291067185
Last Volume Bytes:      1,505,129,472 (1.505 GB)
Non-fatal FD errors:    0
SD Errors:              0
FD termination status:  OK
SD termination status:  OK
Termination:            Backup OK
```

Acima podemos perceber que o *backup* foi realizado à taxa de 6,9 MB/s de velocidade de transferência do módulo nice-sd para gravação na fita LTO-1. Esta taxa é resultado da razão entre a quantidade total em MB do *backup* e o tempo que transitou pelo respectivo módulo. Caso o *job* ficasse retido esperando que um volume fosse nomeado, o tempo transcorrido no Director aumentaria, causando uma diminuição da taxa neste módulo. Logo a taxa de velocidade é relativa para cada módulo.

Alternadamente a cada teste de *backup*, um teste de *restore* era realizado com o *job* RestoreFiles, para recuperar os arquivos dos *backups* realizados anteriormente. De acordo com o trecho abaixo o atributo “Where” informa que os arquivos serão recuperados para o diretório /tmp/bacula-*restore*.

```
*restore
Automatically selected Catalog: MyCatalog
Using Catalog "MyCatalog"
[...]
Enter JobId(s), comma separated, to restore: 1
You have selected the following JobId: 18
[...]
1 file marked.
[...]
JobName:         RestoreFiles
Bootstrap:       /opt/bacula/working/nice-dir.restore.3.bsr
Where:           /tmp/bacula-restores
Replace:         always
FileSet:         Full_Set
Backup Client:   nice-fd
Restore Client:  nice-fd
Storage:         nice-STKL20-LTO-1
When:            2010-11-29 19:04:08
Catalog:         MyCatalog
Priority:        10
[...]
```

De acordo com o trecho abaixo, a mensagem do *log* informa que o dispositivo padrão

"FileStorage" será usado e que os arquivos foram restaurados a uma taxa de 3,01 MB/s.

```
[...]
*messages
[...]
  Build OS:               i686-pc-linux-gnu redhat
  JobId:                  19
  Job:                    RestoreFiles.2010-11-29_19.04.22_07
  Restore Client:         nice-fd
  Start time:             29-Nov-2010 19:04:24
  End time:               29-Nov-2010 19:11:32
  Files Expected:         1
  Files Restored:         1
  Bytes Restored:         1,288,870,533
  Rate:                   3011.4 KB/s
  FD Errors:              0
  FD termination status:  OK
  SD termination status:  OK
  Termination:            Restore OK
[...]
```

A estrutura original da pasta foi mantida, bem como, os respectivos atributos específicos de permissionamentos e datas de criação/modificação.

```
[...]
/tmp/bacula-restores/:
total 4.0K
drwxr--r-- 2 root root 4.0K Nov 29 19:06 root

/tmp/bacula-restores/root:
total 1.3G
-rw-r--r-- 1 root root 1.3G Nov 29 18:42 arquivo_teste.dmp.gz
[...]
```

As configurações realizadas para adicionar um cliente ou um dispositivo de armazenamento foram, em grande parte, deduzidas dos exemplos comentados nos próprios arquivos de configuração, do manual oficial, [5], e do site [31].

Os testes 3 e 4 do *host* atena, são semelhantes aos testes 1 e 2, com a diferença de que a *storage* adicionada é a IBM TS3200. A Figura 28 esquematiza a realização dos testes 3 e 4.

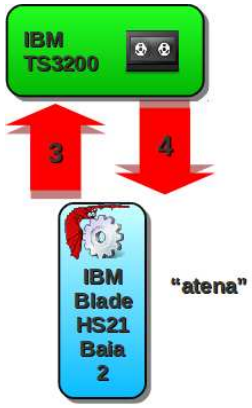


Figura 28. Esquema dos testes 3 e 4.

O trecho abaixo descreve a saída obtida do *backup* realizado com o teste de número 3. Nota-se que a taxa de velocidade geral do *backup* representa aproximadamente 20,97 MB/s. Nenhuma compressão ou encriptação dos dados foi realizada, da mesma forma que o teste 1.

```
[...]
  Build OS:               x86_64-unknown-linux-gnu redhat
  JobId:                  2
  Job:                    BackupClient1.2010-11-29_19.14.13_07
  Backup Level:           Full
  Client:                 "atena-fd" 5.0.3 (04Aug10) x86_64-unknown-linux-gnu,redhat,
  FileSet:                "Teste-Arquivo" 2010-11-29 19:14:13
  Pool:                   "Tape-LTO-3" (From User input)
  Catalog:                "MyCatalog" (From Client resource)
  Storage:                "atena-TS3200-1-LTO-3" (From user selection)
  Scheduled time:         29-Nov-2010 19:13:02
  Start time:             29-Nov-2010 19:14:15
  End time:               29-Nov-2010 19:15:15
  Elapsed time:           1 min
  Priority:               10
  FD Files Written:       1
  SD Files Written:       1
  FD Bytes Written:       1,288,870,533 (1.288 GB)
  SD Bytes Written:       1,288,870,645 (1.288 GB)
  Rate:                   21481.2 KB/s
  Software Compression:   None
  VSS:                    no
  Encryption:             no
[...]
  FD termination status:  OK
  SD termination status:  OK
  Termination:            Backup OK
[...]
```

O teste 4 é a realização de um *job* que restaura a informação gravada pelo *job* de *backup* do teste 3. Pode-se notar que para este *restore* a velocidade 32,27 MB/s.

```
Build OS:               x86_64-unknown-linux-gnu redhat
JobId:                  3
Job:                    RestoreFiles.2010-11-29_19.21.18_08
Restore Client:         atena-fd
Start time:             29-Nov-2010 19:21:20
End time:               29-Nov-2010 19:21:59
Files Expected:         1
Files Restored:         1
Bytes Restored:         1,288,870,533
Rate:                   33048.0 KB/s
FD Errors:              0
FD termination status:  OK
SD termination status:  OK
Termination:            Restore OK
```

Todos os quatro testes foram bem sucedidos no propósito de testar os recursos mais básicos de *backup* e *restore* do Bacula. Porém, submetendo os testes à uma breve análise dos resultados, uma característica chama atenção: a reduzida taxa de velocidade de manipulação da fita em comparação com os valores teóricos divulgados na Tabela 2, no início do capítulo. O acompanhamento da taxa de velocidade pode indicar problemas na estratégia de *backup*, principalmente em relação às configurações utilizadas, distribuição dos módulos ou tráfego da rede.

Desta forma o *btape* foi novamente utilizado, desta vez, para executar testes de velocidade e de preenchimento de fita. Este teste com a *btape* ocorreu em âmbito local, tal qual a execução dos testes anteriores. Logo, a influência da rede Ethernet está descartada em todos os testes.

O *btape*, utilizado com o comando *speed* executa testes de velocidade. As Figuras 29 e 30, no Apêndice B, mostram o resultado destes testes de velocidade. Por padrão, o *btape* transfere para fita fluxos de dados que variam em tamanho de 1, 2 e 4 GB, gravados em blocos de 64 KB de tamanho. Medições para 8, 16 e 32 GB também foram feitos, visto que na realidade da empresa há arquivos com tamanhos semelhantes. Também no Apêndice B, podem ser conferidos os resultados de mais testes.

Os testes abrangem o preenchimento da fita com fluxo de dados de valor zero, testes com fluxo de dados de valores randômicos, e testes com a escrita de estruturas de blocos no formato de gravação do Bacula. Segundo o manual, [5], quando a compressão de dados é utilizada pela biblioteca de fitas, o teste aleatório determinará a menor vazão de dados do *drive*. Quando *strings* constantes são usados, como um fluxo de zeros, o teste determinará a máxima velocidade da cadeia de *hardwares* envolvidos, que são: a CPU, a memória, a interface SCSI, o cabo, o *drive* e a fita. O disco rígido não é mencionado, porque o *hardware* que transmite fluxos de dados zero e valores aleatórios é a CPU.

Quando os testes 1, 2, 3 e 4 são comparados às taxas de velocidades apresentadas pelas figuras acima, nota-se que o comportamento dos *backups* e *restores* de arquivos reais aproxima-se mais ao teste de velocidade com dados randômicos. Os testes randômicos representam a menor vazão de gravação dos *drives* de fita por causa dos algoritmos de escrita física de dados na fita. Quanto maior a quantidade de blocos de dados brutos com valores constantes, maior é a eficiência do algoritmo na escrita das fitas. O que reforça mais ainda esta característica, é a gravação de dados previamente compactados à nível de *software*. Desta forma, o recurso de compactação em nível de *hardware* é desativado, pois não mais a possibilidade dos dados serem recompactados.

Outros dois fatores que justificam a redução da velocidade dos *backups* e *restores* para valores abaixo de 50% da velocidade ideal são:

- Influência das taxas de velocidades dos discos rígidos, uma vez que o *btape* transfere seu fluxo de dados diretamente da CPU, assim como os testes que utilizam o utilitário dd com os pseudo-dispositivos */dev/zero* e */dev/random*;

- O ajuste do tamanho dos blocos de dados através de diretivas no recurso que define o objeto *Device*, para otimizar tanto a transferência em rede como também para gravação em fitas.

Capítulo 7
Conclusão e Trabalhos Futuros

7.1. Conclusão

As empresas estão ávidas por consumir as tecnologias de ponta oferecidas pelo mercado, mas não possuem maturidade suficiente no trato com os *softwares* e equipamentos recém-adquiridos. Desta maneira, os administradores de sistemas acabam por se adaptar às soluções de seus fornecedores e parceiros, sem um prévio estudo de viabilidade, de planejamento e monitoramento de *performance* de recursos.

O presente trabalho teve como principal motivação questões técnicas reais, de uma empresa pública de médio porte, relacionadas à questão de *backup* corporativo. Outras motivações foram a popularidade e estatísticas, creditadas pelo Sourceforge, a sistemas de *backup* corporativo *open source* e a divulgações e relatos de comunidades do mundo do *software* livre. Como resultado, foi proposto o estudo de uma opção de *software open source* para sistemas de *backup*, o Bacula, que traz além da economia obtida com seu uso, a sua solução bastante competitiva frente às mais modernas soluções comerciais do mercado. Este trabalho, porém, não reflete o posicionamento da empresa na qual o estudo foi realizado, uma vez que a mesma está atrelada a uma solução de *backup* comercial, não havendo, portanto, autorização oficial no uso e publicação das informações internas. Por este motivo o nome da empresa foi mantido em sigilo.

Os resultados obtidos em torno dos requisitos levantados procuraram obedecer a critérios mínimos específicos da empresa, e à infraestrutura computacional que ela suporta. Apesar da empresa estar comprometida com uma série de restrições contratuais, isto não deve caracterizar um impedimento aos estudos de compatibilidade, funcionalidade e *performance* com base em um cenário de equipamentos corporativos reais, para avaliar a possibilidade de se ter um "plano B" para realizar *backups*. Afinal, o cumprimento da função básica de um *backup* deve ser garantido como meio de preservar a vida útil de informações e restaurá-las quando preciso.

Surgiram muitas dificuldades na realização deste trabalho, que exigiram maior dedicação no estudo da infraestrutura, composta por modernas *blades*, *storages*, *switchs* e bibliotecas de fitas, da IBM, adquiridas à apenas 1 ano, bem como à complexidade do próprio estudo de viabilidade.

Em relação à infraestrutura, pode-se citar:

- O *switch* FC não estava devidamente configurado para integrar as bibliotecas de fitas TS3200 com o servidor de backup, localizado na *blade* 2. Este processo de integração baseia-se no processo chamado de zoneamento, onde são realizadas configurações que interconectam pares de dispositivos através da criação de zonas dentro de uma *vsan*. Foi comprovado, na prática, que a realização de zoneamento em sistemas onde não há

redundância de *switchs* pode provocar desconexão de dispositivos, resultando, por exemplo, em quedas de máquinas virtuais e de seus respectivos serviços, uma vez que seus SOs e armazenamentos de dados localizam-se na *storage* de discos.

- O *software* IBM Tivoli Storage Manager (TSM) versão 6.2 foi adquirido, pela empresa, e implantado na *blade 2* como sistema principal de *backup*. Consequentemente, o SO RedHat Enterprise 5.4 foi instalado por questões de obediência a requisitos de suporte e manutenção do *software* TSM por uma empresa contratada. Esta instalação causou a perda do SO CentOS 5.5 para testes com o Bacula, pois todo o espaço dos discos rígidos internos da *blade* foi utilizado. Desta forma, o CentOS foi reinstalado em discos FC para compartilhar um *dual-boot* com o SO principal.

Em relação ao estudo de viabilidade, não houve a possibilidade de apresentar testes mais abrangentes e com maior número de combinações possíveis de configurações do sistema de *backup*. O grande número de combinações tende a crescer exponencialmente com o aumento do número de variáveis. Por exemplo, se os testes visarem a configuração do recurso Storage com o objetivo de medir a taxa de velocidade dos *jobs* de *backups* de acordo com diferentes mídias de armazenamento, ter-se-á pelo menos testes D2D e D2T, fora a combinação dos valores que suas diretivas podem assumir, como *"Media Type"* que pode ser tanto LTO-3 quanto LTO-4, na biblioteca de fitas IBM TS3200. Desta maneira, de um conjunto de *n* diretivas escolhidas para os testes serem realizados, se cada uma assumir, no mínimo, apenas dois valores, ter-se-á 2^n testes. Seria exaustiva a realização manual de semelhantes testes. Para tentar contornar o problema, *scripts* de automatização poderiam ter sido criados. Porém, grande parte do tempo seria despendido no desenvolvimento de *scripts*, o que também constituiria fuga ao escopo.

Através deste trabalho foi possível ter uma maior experiência de uso de sistemas de *backup* e ao mesmo tempo apresentar lições aprendidas com um estudo de viabilizade de uso do Bacula, ainda que restrito, considerando problemas de um ambiente corporativo real. Entre os requisitos que puderam ser vivenciados ao longo do estudo, estão: o gerenciamento centralizado, capacidades disponíveis para restauração e verificação de dados, integridade e confiabilidade dos dados copiados dos clientes para suas oportunas restaurações , escalabilidade e vazão.

O Bacula mostrou ser compatível com a infraestrutura de *hardware* dos dois ambientes propostos inicialmente, apresentando um bom desempenho nas funcionalidades básicas necessárias em um sistema de *backup* corporativo. Portanto conclui-se que o *software* Bacula é viável por atender de um modo geral, em termos de custo/benefício, aos requisitos técnicos e financeiros da empresa.

7.2. Trabalhos Futuros

Podem ser citados como sugestões de trabalhos futuros:

- Análise de *benchmark* de *hardwares* e sistemas que pertencem ao fluxo de dados, de forma a determinar as melhores estratégias de *backup* e *restore*;

- Ampliar a variabilidade de configurações de testes, para realizar uma análise mais abrangente do Bacula;
- Realizar um estudo comparativo entre ferramentas de *software* livre e proprietário de *backup*;
- Contribuir com o desenvolvimento do Bacula, sobretudo as interfaces gráficas. O aprimorar a usabilidade Bacula com a construção de uma interface gráfica mais completa e sustentável;
- Estudo para implantação de servidores redundantes de *backup*, cujo catálogo de informações esteja localizado num SGBD em cluster.

Referências Bibliográficas

[1] IBM Corp. IBM Tivoli Storage Manager Version 6.2 information center. Disponível em: <http://publib.boulder.ibm.com/infocenter/tsminfo/v6r2/index.jsp> Acesso em: 28 nov. 2010.

[2] Computer Associates, Inc. Data Protection Solutions - Backup, Disk to Disk, Replication, High Availability. Disponível em: <http://arcserve.com/us/> Acesso em: 28 nov. 2010.

[3] Sourceforge. Disponível em: <http://sourceforge.net/project/stats/detail.php?group_id=50727&ugn=bacula&type=prdownload&mode=alltime&file_id=0> Acesso em: 29 nov. 2010.

[4] Bacula, the Open Source, Enterprise ready, Network Backup Tool for Linux, Unix, Mac and Windows. Disponível em: <http://www.bacula.org/>. Acesso em: 29 nov. 2010.

[5] Bacula. Bacula Main Reference Guide. Disponível em: <http://www.bacula.org/5.0.x - manuals/en/ main/main/index.html> Acesso em: 29 nov. 2010.

[6] Wikipedia. Backup. Disponível em: <http://pt.wikipedia.org/wiki/Backup> Acesso em: 29 nov. 2010.

[7] Wikipedia. RAID. Disponível em: <http://pt.wikipedia.org/wiki/RAID> Acesso em: 29 nov. 2010.

[8] Remote Data Backups. RAID Array Backup. Disponível em: <http://www.remote databackups.com/why/backup-raid-drive.cfm> Acesso em: 29 nov. 2010.

[9] LEAVER, Michael J.. RAID is Not a Backup Solution. 2BrightSparks Pte Ltd, 2004-2007.

[10] VERAS, Manoel. Datacenter 1.0. Disponível em: <http://datacenter10.blogspot.com/> Acesso em: 29 nov. 2010.

[11] REINE, David; KAHN, Mike. Disk and Tape Square Off Again — Tape Remains

King of the Hill with LTO-4. Clipper Notes, 13 fev. 2008.

[12] MCADAM, Dianne. The Truth about Tape — Nine Myths to Reconsider. Clipper Notes, 21 fev. 2007.

[13] Bacula DokuWiki. Bacula vs Other Backup Solutions. Disponível em: <http://wiki.bacula.org/doku.php?id=comparisons> Acesso em: 29 nov. 2010.

[14] PRESTON, W. Curtis. Open-Source Data Backup Options for Your Data Center. Disponível em: <http://searchdatabackup.techtarget.com/generic/0,295582,sid187_gci1362372,00.htm>. Acesso em: 27 nov. 2010.

[15] HEXSEL, Roberto A. Software Livre – Propostas de Ações de Governo para Incentivar o Uso de Software Livre. Curitiba: 2002. 49p. Disponível em: <http://www.inf.ufpr.br/info/techrep/RT_DINF004_2002.pdf>. Acesso em: 30 nov. 2010.

[16] Com ciência. O Software Livre está dentro da lei?. Disponível em: <http://www.comciencia.br/200406/reportagens/06.shtml> Acesso em: 25 nov. 2010.

[17] CABRAL, Bruna. Uso de software livre vira lei na Prefeitura do Recife. Jornal do Comércio - Recife. Disponível em: <http://www.bfsf.it/legislazione/brasile-recife.htm> Acesso em: 25 nov. 2010.

[18] PRESTON, W. Curtis. BACKUPCENTRAL. Disponível em: <http://www.backupcentral .com>. Acesso em: 26 nov. 2010.

[19] Dell Inc. Dell PowerEdge 2600 Server. Disponível em: <http://www.dell.com/downloads/global/products/pedge/en/2600_specs.pdf> Acesso em: 30 nov. 2010.

[20] Sun Microsystems, Inc. L20 Tape Library User's Guide. Disponível em: <http://dlc.sun.com/pdf/CRCM2235/CRCM2235.pdf> Acesso em: 28 nov. 2010.

[21] IBM Corp. IBM BladeCenter® HS21. Disponível em: <http://www.ibm.com/br/systems/bladecenter/hs21/index.phtml> Acesso em: 27 nov. 2010.

[22] IBM Corp. Biblioteca de Fitas TS3200 Modelo Express. Disponível em: <http://www.ibm.com/br/systems/storage/tape/ts3200/index.phtml>. Acesso em: 22

nov. 2010.

[23] Wikipedia. Linear Tape Open. Disponível em: <http://en.wikipedia.org/wiki/Linear_Tape-Open> Acesso em: 22 set. 2010.

[24] Fibre Channel Industry Association. FCIA Roadmap Useful Documents and Resources. Disponível em: <http://www.fibrechannel.org/roadmaps> Acesso em: 22 nov. 2010.

[25] IBM Corp. IBM System Storage TS3200 Tape Library. 2007. Disponível em: <http://www.arp.com/medias/pdf/MIME033774.pdf > Acesso em: 30 nov. 2010.

[26] Bacula DokuWiki. FAQ. Disponível em: <http://wiki.bacula.org/doku.php?id=faq> Acesso em: 30 nov. 2010.

[27] WikiVS, the open comparison website. MySQL vs. PostgreSQL. Disponível em: <http://www.wikivs.com/wiki/MySQL_vs_PostgreSQL> Acesso em: 30 nov. 2010.

[28] HUNTER, Shylon Ray. MySQL vs. PostgreSQL. TechRepublic, 2002. Disponível em: <http://articles.techrepublic.com.com/5100-10878_11-1050671.html> Acesso em: 30 nov. 2010.

[29] Bacula Systems. Enterprise Backup and Support. Disponível em: <http://www.baculasystems.com> Acesso em: 30 nov. 2010.

[30] Wikipedia. Fibre-Channel. Disponível em: <http://en.wikipedia.org/wiki/Fibre-Channel> Acesso em: 30 out. 2010.

[31] Bacula DokuWiki. Sample Configs. Disponível em: <http://wiki.bacula.org/doku.php?id =sample_configs> Acesso em: 30 nov. 2010.

[32] TI INSIDE Online. Aprovado projeto que prioriza compra software livre na administração pública. Disponível em: <http://www.tiinside.com.br/24/11/2010/aprovado-projeto-que-prioriza-compra-software-livre-na-administracao-publica/ti/ 205095/news.aspx> Acesso em: 30 nov. 2010.

[33] JusBrasil Diários. Pg. 43. Seção 3. Diário Oficial da União (DOU) de 04/04/2006. Disponível em: <http://www.jusbrasil.com.br/diarios/354739/dou-secao-3-04-04-2006-pg-43> Acesso em: 30 nov. 2010.

[34] JusBrasil Diários. Pg. 2. Seção 3. Diário Oficial da União (DOU) de 03/08/2010. Disponível em: <http://www.jusbrasil.com.br/diarios/7190742/dou-secao-3-03-08-2010-pg-2> Acesso em: 30 nov. 2010.

[35] Ultrium Lto3. Disponível em: <http://www.ultrium.com>.Acesso em: 22 set. 2010.

[36] PRESTON, W. Curtis. Backup & Recovery. Sebastopol, Ca: O'Reilly Media, Inc., 2006. Disponível em: <http://books.google.com/books?id=6-w4fXbBInoC&>. Acesso em: 9 set. 2010.

[37] GUISE, Preston De. Enterprise Systems Backup and Recovery: A Corporate Insurance Policy. Boca Raton, Fl: Auerbach Publication, 2009.

[38] LITTLE, David B.; CHAPA, David A.. Implementing Backup and Recovery: The Readiness Guide for the Enterprise. Indianapolis: Wiley Publishing, Inc., 2003.

[39] Fundação do Software Livre. Disponível em: <http://www.fsf.org>.Acesso em: 22 set. 2010.

[40] CARDOSO, Rodrigo Maluf; ZIZIOTTI, Fernando B.B.; Daniel Gama e Colombo. Software livre no Brasil. Disponível em: <http://www.migalhas.com.br/mostra_noticia_articuladas.aspx?cod=3560> Acesso em: 30 out. 2010

[41] IBM Corp. Biblioteca de Fitas TS3200 Modelo Express. Disponível em: <http://www.ibm.com/br/systems/storage/tape/ts3200/index.phtml>. Acesso em: 22 set. 2010.

[42] FindIcons.com. Icon Search Engine. Disponível em: <http://findicons.com/> 2010. Acesso em: 29 nov. 2010.

Glossário

Em ordem alfabética:

- **Atributos de arquivo:** Os atributos de arquivo são informações para identificar um arquivo e todas as suas propriedades, como tamanho, data de criação, data de modificação, permissões, etc. Normalmente, os atributos são suportados integralmente pelo Bacula. Os atributos não incluem os dados do arquivo.
- **Autochanger ou Autoloader:** dispositivo de armazenamento de dados consistindo em ao menos uma unidade de fita (o acionador, ou *drive*), um método de carregamento das fitas no *drive* (o robô) e uma área de armazenamento para as fitas (as gavetas, ou *magazines*). Autoloaders maiores com vários *drives*, robôs e magazines são conhecidos como bibliotecas de fitas.
- ***Backup*:** O termo refere-se a um *job* para salvar os arquivos do Bacula.
- **Catálogo:** O catálogo é usado para armazenar informações de resumo sobre os *jobs*, os clientes e arquivos que foram copiados e em que volume ou volumes. A informação guardada no catálogo permite que o administrador ou o utilizador determine quais trabalhos foram executados, o seu estatuto, bem como as características mais importantes de cada arquivo que foi feito o *backup*, e mais importante, permite a escolha de arquivos para restaurar.
- **Cliente:** Refere-se à máquina que onde está sendo feito o *backup*, e é sinônimo de serviços de arquivos ou File Daemon, e muitas vezes, é referido como o FD. Um cliente é definido como um recurso de arquivo de configuração.
- **Console:** O programa que faz interface com o director permitindo que o usuário ou administrador de sistema possa controlar o Bacula.
- **Daemon:** Terminologia Unix para um programa que está sempre presente em segundo plano para realizar uma tarefa designada. Em sistemas Windows, bem como alguns sistemas Unix, daemons são chamados de serviços.
- **Diretiva:** O termo diretiva é usado para se referir a uma declaração ou um registro em um recurso em um arquivo de configuração, definindo uma configuração específica.

Por exemplo, a directiva "*Name"* define o nome do recurso.

- **Disaster Recovery:** Plano de recuperação de um sistema depois de ter acontecido um desastre natural ou induzido pelo homem, inviabilizando o sistema. Este procedimento visa a recuperação do acesso a dados, comunicações e áreas de trabalho.
- ***Drive*:** Unidade de armazenamento ou de leitura de dados, pertencente ao *hardware* de um computador.
- **File Daemon:** O *daemon* em execução no computador cliente para fazer *backup*. Este é também referido como os serviços de arquivo, e às vezes como os serviços de cliente ou o DF.
- **FileSet:** É um recurso contido em um arquivo de configuração que define os arquivos a serem incluídos no *backup*. É constituída por uma lista de arquivos/diretórios incluídos, uma lista de arquivos/diretórios excluídos, e como o arquivo será armazenado (compressão, criptografia, assinatura).
- **Incremental:** Um *backup* que inclui todos os arquivos alterados desde a última completo, diferencial ou *backup* incremental iniciado. Ele normalmente é especificada sobre a diretiva **de nível** dentro da definição de recursos de trabalho, ou em uma agenda de recursos.
- **Janela de *backup*:** Período do dia reservado para *backup* de dados.
- **Job:** Um *job* Bacula é um recurso de configuração que define o trabalho que Bacula deve executar para fazer *backup* ou restaurar a partir de um determinado cliente. Trata-se do tipo (*backup*, restauração, verificação, etc), o nível (completo, incremental,...), o conjunto de arquivos e o dispositivo de armazenamento onde os arquivos devem ser copiados.
- **LAN:** Refere-se a um grupo de computadores e dispositivos associados que compartilham uma linha de comunicação comum ou conexão sem fio. Normalmente, os dispositivos conectados compartilham os recursos de um único processador ou servidor em uma pequena área geográfica (por exemplo, num edifício de escritórios).
- **Largura de banda:** Em redes de computadores, a largura de banda é muitas vezes usada como sinônimo de taxa de transferência de dados - a quantidade de dados que pode ser transportada de um ponto a outro em um determinado período de tempo (geralmente um segundo). Esse tipo de largura de banda é normalmente expressa em

bits (de dados) por segundo (bps). Ocasionalmente, é expresso em bytes por segundo (Bps).

- **Monitor:** O programa que faz interface com todos os daemons permitindo que o usuário ou administrador do sistema monitore o *status* do Bacula.
- **NAS:** *Network-Attached Storage* é um dispositivo dedicado ao armazenamento de arquivos dentro de uma rede, provendo acesso heterogêneo aos dados para os clientes desta rede. O dispositivo de armazenamento anexado à rede é conectado a uma rede de área local (normalmente, uma rede Ethernet) e é atribuído um endereço IP. Os Arquivos requisitados são mapeadas pelo servidor principal para o servidor de arquivos NAS. O armazenamento ligado à rede consiste de armazenamento em disco rígido, incluindo sistemas de multi-discos RAID e *software* de configuração e mapeamento de locais de arquivo para o dispositivo ligado à rede.
- **Off-Site:** É a estratégia de envio de dados críticos para fora do local principal onde é feito o backup, como parte de um plano de recuperação de desastres.
- **Período de retenção:** Existem vários tipos de períodos de retenção que o Bacula reconhece. Os mais importantes são os períodos de retenção de arquivos, períodos de retenção de *jobs*, e períodos de retenção de volumes. Para cada um desses períodos de retenção está relacionado com o tempo em que registros específicos serão mantidos na base de dados do catálogo. Isso não deve ser confundido com o tempo que os dados em um volume são válidos. O períodos de retenção de arquivos, por exemplo, determina o tempo que os registros de arquivos são mantidos no banco de dados de catálogo. Este período é importante por duas razões: a primeira é que, enquanto os registros do arquivo permanecer no banco de dados, você pode "ver" o banco de dados via um programa de console e restaurar qualquer arquivo individual. Uma vez que os registros de arquivo são removidos de um banco de dados, os arquivos individuais de um *job* de *backup* não podem continuar a serem consultados. A segunda razão para escolher cuidadosamente o período de retenção dos arquivos é porque o volume de registros no banco de dados de arquivo usa mais espaço de armazenamento no banco de dados. Como conseqüência, deve ser garantido que a poda "regular" dos registros do arquivo de banco de dados seja feita para manter o banco de dados longe de um crescimento muito grande.

- **Pool:** Um cojunto de recursos inicializados que são mantidos para uso.
- **Recurso:** É uma parte de um arquivo de configuração que define uma determinada unidade de informação que está disponível para Bacula. É constituída de várias directivas (instruções de configuração individual). Por exemplo, o recurso do *job* define todas as propriedades de uma tarefa específica: nome, horário, volume da pool, tipo de *backup*, o nível de *backup*, ...
- ***Restore*:** A restauração é um recurso de configuração que descreve a operação de recuperação de um arquivo de mídia de *backup*. É o inverso de uma gravação, só que na maioria dos casos, a restauração terá normalmente um pequeno conjunto de arquivos a serem restaurados, enquanto que normalmente um *backup* salva todos os arquivos no sistema. Claro que, depois de uma falha no disco, o Bacula pode ser chamado a fazer uma restauração completa de todos os arquivos que estavam no sistema.
- **SAN:** Um SAN é uma rede ou sub-rede de alta velocidade para fins especiais que interliga diferentes tipos de dispositivos de armazenamento de dados com os servidores de dados associados, em nome de uma rede maior de usuários. Um SAN consiste em uma infra-estrutura de comunicação que provê conexões físicas com uma camada de gerenciamento, que organiza as conexões, os dispositivos de armazenamento e os computadores, tornando a transferência de dados robusta e segura.
- **Scan:** Uma operação de scan faz com que o conteúdo de um volume ou uma série de volumes sejam verificados. Estes volumes com as informações sobre os arquivos que eles contêm são restauradas para o catálogo do Bacula. Assim que a informação é restaurada para o catálogo, os arquivos contidos nesses volumes podem ser facilmente recuperados. Esta função é particularmente útil se determinados volumes ou *jobs* tenham excedido o período de retenção e foram podados ou removidos do catálogo. A digitalização a partir de volumes de dados para o catálogo é feita usando o programa bscan.
- **Schedule:** É um recurso de configuração que define quando o *job* será agendado para execução. Para utilizar o schedule, o recurso do *job* fará referência ao respectivo nome de agendamento.

- **Script:** Um arquivo de instruções para a execução de tarefas específicas.
- **Serviço:** Este é um programa que fica permanentemente em memória aguardando instruções. Em ambientes Unix, os serviços também são conhecidos como **daemons.**
- **Slot:** Espaço nas bibliotecas de fitas onde são colocados os cartuchos de fitas.
- **Storage Daemon:** O *daemon* de armazenamento, por vezes referido como o SD, é o código que grava os atributos e os dados para um volume de armazenamento (geralmente uma fita ou disco).
- **Tape Library:** Biblioteca de fitas, também chamada *storage* de fitas. Uma rede de área de armazenamento pode usar tecnologia de comunicação existentes, como FC.
- **Volume:** Um volume é uma unidade de arquivo, normalmente, uma fita ou um arquivo de disco chamado por Bacula, onde armazena os dados de um ou mais *jobs* de *backup*. Todos os Volumes do Bacula devem ter um label para identificá-lo.
- **VPN:** Uma rede virtual privada é uma rede que utiliza uma infraestrutura pública de telecomunicações, como a Internet, para fornecer aos usuários acesso remoto seguro à rede interna da sua organização. A VPN mantém a privacidade através de procedimentos de segurança e protocolos de tunelamento, como o L2TP.

Apêndice A:
Testando as bibliotecas de fitas com o utilitário *btape*

Host nice – fita LTO-1 carregada no _drive_ 0 da StorageTek L20:

```
[root@nice bin]# mtx status

  Storage Changer /dev/changer:1 Drives, 11 Slots ( 1 Import/Export )

Data Transfer Element 0:Full (Storage Element 1 Loaded):VolumeTag = LTO029
      Storage Element 1:Empty
      Storage Element 2:Full :VolumeTag=LTO022
      Storage Element 3:Full :VolumeTag=LTO096
      Storage Element 4:Full :VolumeTag=LTO093
      Storage Element 5:Full :VolumeTag=LTO086
      Storage Element 6:Full :VolumeTag=LTO010
      Storage Element 7:Full :VolumeTag=LTO017
      Storage Element 8:Full :VolumeTag=LTO045
      Storage Element 9:Full :VolumeTag=LTO052
      Storage Element 10:Full :VolumeTag=LTO067
      Storage Element 11 IMPORT/EXPORT:Empty
[root@nice bin]# ./btape -c ../etc/bacula-sd.conf /dev/nst0
Tape block granularity is 1024 bytes.
btape: butil.c:284 Using device: "/dev/nst0" for writing.
15-Nov 11:37 btape JobId 0: 3301 Issuing autochanger "loaded? drive 0" command.
15-Nov 11:37 btape JobId 0: 3302 Autochanger "loaded? drive 0", result is Slot 1.
btape: btape.c:476 open device "LTO-1" (/dev/nst0): OK
*test

=== Write, rewind, and re-read test ===

I'm going to write 10000 records and an EOF
then write 10000 records and an EOF, then rewind,
and re-read the data to verify that it is correct.

This is an *essential* feature ...

btape: btape.c:1148 Wrote 10000 blocks of 64412 bytes.
btape: btape.c:608 Wrote 1 EOF to "LTO-1" (/dev/nst0)
btape: btape.c:1164 Wrote 10000 blocks of 64412 bytes.
btape: btape.c:608 Wrote 1 EOF to "LTO-1" (/dev/nst0)
btape: btape.c:1206 Rewind OK.
```

```
10000 blocks re-read correctly.
Got EOF on tape.
10000 blocks re-read correctly.
=== Test Succeeded. End Write, rewind, and re-read test ===

btape: btape.c:1274 Block position test
btape: btape.c:1286 Rewind OK.
Reposition to file:block 0:4
Block 5 re-read correctly.
Reposition to file:block 0:200
Block 201 re-read correctly.
Reposition to file:block 0:9999
Block 10000 re-read correctly.
Reposition to file:block 1:0
Block 10001 re-read correctly.
Reposition to file:block 1:600
Block 10601 re-read correctly.
Reposition to file:block 1:9999
Block 20000 re-read correctly.
=== Test Succeeded. End Write, rewind, and re-read test ===



=== Append files test ===

This test is essential to Bacula.

I'm going to write one record  in file 0,
                 two records in file 1,
           and three records in file 2

btape: btape.c:578 Rewound "LTO-1" (/dev/nst0)
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:608 Wrote 1 EOF to "LTO-1" (/dev/nst0)
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:608 Wrote 1 EOF to "LTO-1" (/dev/nst0)
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:608 Wrote 1 EOF to "LTO-1" (/dev/nst0)
btape: btape.c:476 open device "LTO-1" (/dev/nst0): OK
btape: btape.c:578 Rewound "LTO-1" (/dev/nst0)
```

```
btape: btape.c:1418 Now moving to end of medium.
btape: btape.c:629 Moved to end of medium.
We should be in file 3. I am at file 3. This is correct!

Now the important part, I am going to attempt to append to the tape.

btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:608 Wrote 1 EOF to "LTO-1" (/dev/nst0)
btape: btape.c:578 Rewound "LTO-1" (/dev/nst0)
Done appending, there should be no I/O errors

Doing Bacula scan of blocks:
1 block of 64448 bytes in file 1
End of File mark.
2 blocks of 64448 bytes in file 2
End of File mark.
3 blocks of 64448 bytes in file 3
End of File mark.
1 block of 64448 bytes in file 4
End of File mark.
Total files=4, blocks=7, bytes = 451,136
End scanning the tape.
We should be in file 4. I am at file 4. This is correct!

The above Bacula scan should have output identical to what follows.
Please double check it ...
=== Sample correct output ===
1 block of 64448 bytes in file 1
End of File mark.
2 blocks of 64448 bytes in file 2
End of File mark.
3 blocks of 64448 bytes in file 3
End of File mark.
1 block of 64448 bytes in file 4
End of File mark.
Total files=4, blocks=7, bytes = 451,136
=== End sample correct output ===

If the above scan output is not identical to the
sample output, you MUST correct the problem
or Bacula will not be able to write multiple Jobs to
the tape.


=== Write, backup, and re-read test ===

I'm going to write three records and an EOF
then backup over the EOF and re-read the last record.
```

```
Bacula does this after writing the last block on the
tape to verify that the block was written correctly.

This is not an *essential* feature ...

btape: btape.c:578 Rewound "LTO-1" (/dev/nst0)
btape: btape.c:813 Wrote first record of 64412 bytes.
btape: btape.c:824 Wrote second record of 64412 bytes.
btape: btape.c:835 Wrote third record of 64412 bytes.
btape: btape.c:608 Wrote 1 EOF to "LTO-1" (/dev/nst0)
btape: btape.c:851 Backspaced over EOF OK.
btape: btape.c:856 Backspace record OK.
btape: btape.c:874
Block re-read correct. Test succeeded!
=== End Write, backup, and re-read test ===



=== Forward space files test ===

This test is essential to Bacula.

I'm going to write five files then test forward spacing

btape: btape.c:578 Rewound "LTO-1" (/dev/nst0)
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:608 Wrote 1 EOF to "LTO-1" (/dev/nst0)
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:608 Wrote 1 EOF to "LTO-1" (/dev/nst0)
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:608 Wrote 1 EOF to "LTO-1" (/dev/nst0)
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:608 Wrote 1 EOF to "LTO-1" (/dev/nst0)
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:608 Wrote 1 EOF to "LTO-1" (/dev/nst0)
btape: btape.c:578 Rewound "LTO-1" (/dev/nst0)
```

```
btape: btape.c:1632 Now forward spacing 1 file.
We should be in file 1. I am at file 1. This is correct!
btape: btape.c:1644 Now forward spacing 2 files.
We should be in file 3. I am at file 3. This is correct!
btape: btape.c:578 Rewound "LTO-1" (/dev/nst0)
btape: btape.c:1657 Now forward spacing 4 files.
We should be in file 4. I am at file 4. This is correct!

btape: btape.c:1675 Now forward spacing 1 more file.
We should be in file 5. I am at file 5. This is correct!

=== End Forward space files test ===


Ah, I see you have an autochanger configured.
To test the autochanger you must have a blank tape
that I can write on in Slot 1.

Do you wish to continue with the Autochanger test? (y/n): y


=== Autochanger test ===

3301 Issuing autochanger "loaded" command.
Slot 1 loaded. I am going to unload it.
3302 Issuing autochanger "unload 1 0" command.
unload status=OK 0
3303 Issuing autochanger "load 1 0" command.
3303 Autochanger "load 1 0" status is OK.
btape: btape.c:476 open device "LTO-1" (/dev/nst0): OK
btape: btape.c:1562 Rewound "LTO-1" (/dev/nst0)
btape: btape.c:1569 Wrote EOF to "LTO-1" (/dev/nst0)

The test autochanger worked!!
*quit
```

***Host* atena – Fitas LTO-3 e LTO-4 carregadas, respectivamente, nos *drive* 0 e 1, da IBM TS3200:**

```
[root@atena bin]# mtx -f /dev/sg7 status
 Storage Changer /dev/sg7:2 Drives, 47 Slots ( 3 Import/Export )
Data Transfer Element 0:Full (Storage Element 1 Loaded)
Data Transfer Element 1:Full (Storage Element 3 Loaded):VolumeTag = 586AESL4

      Storage Element 1:Empty
      Storage Element 2:Full
      Storage Element 3:Empty
      Storage Element 4:Full :VolumeTag=585AESL4
      Storage Element 5:Empty
```

```
      Storage Element 6:Empty
      Storage Element 7:Empty
      Storage Element 8:Empty
      Storage Element 9:Empty
      Storage Element 10:Empty
      Storage Element 11:Empty
      Storage Element 12:Empty
      Storage Element 13:Empty
      Storage Element 14:Empty
      Storage Element 15:Empty
      Storage Element 16:Empty
      Storage Element 17:Empty
      Storage Element 18:Empty
      Storage Element 19:Empty
      Storage Element 20:Empty
      Storage Element 21:Empty
      Storage Element 22:Empty
      Storage Element 23:Empty
      Storage Element 24:Empty
      Storage Element 25:Empty
      Storage Element 26:Empty
      Storage Element 27:Empty
      Storage Element 28:Empty
      Storage Element 29:Empty
      Storage Element 30:Empty
      Storage Element 31:Empty
      Storage Element 32:Empty
      Storage Element 33:Empty
      Storage Element 34:Empty
      Storage Element 35:Empty
      Storage Element 36:Empty
      Storage Element 37:Empty
      Storage Element 38:Empty
      Storage Element 39:Empty
      Storage Element 40:Empty
      Storage Element 41:Empty
      Storage Element 42:Empty
      Storage Element 43:Empty
      Storage Element 44:Empty
      Storage Element 45 IMPORT/EXPORT:Empty
      Storage Element 46 IMPORT/EXPORT:Empty
      Storage Element 47 IMPORT/EXPORT:Empty

[root@atena bin]# ./btape -c ../etc/bacula-sd.conf /dev/nst3
Tape block granularity is 1024 bytes.
btape: butil.c:284 Using device: "/dev/nst3" for writing.
21-Nov 15:27 btape JobId 0: 3301 Issuing autochanger "loaded? drive 0" command.
21-Nov 15:27 btape JobId 0: 3302 Autochanger "loaded? drive 0", result is Slot 1.
btape: btape.c:476 open device "TS3200-1-Drive-0" (/dev/nst3): OK
```

```
*test

=== Write, rewind, and re-read test ===

I'm going to write 10000 records and an EOF
then write 10000 records and an EOF, then rewind,
and re-read the data to verify that it is correct.

This is an *essential* feature ...

btape: btape.c:1148 Wrote 10000 blocks of 64412 bytes.
btape: btape.c:608 Wrote 1 EOF to "TS3200-1-Drive-0" (/dev/nst3)
btape: btape.c:1164 Wrote 10000 blocks of 64412 bytes.
btape: btape.c:608 Wrote 1 EOF to "TS3200-1-Drive-0" (/dev/nst3)
btape: btape.c:1206 Rewind OK.
10000 blocks re-read correctly.
Got EOF on tape.
10000 blocks re-read correctly.
=== Test Succeeded. End Write, rewind, and re-read test ===

btape: btape.c:1274 Block position test
btape: btape.c:1286 Rewind OK.
Reposition to file:block 0:4
Block 5 re-read correctly.
Reposition to file:block 0:200
Block 201 re-read correctly.
Reposition to file:block 0:9999
Block 10000 re-read correctly.
Reposition to file:block 1:0
Block 10001 re-read correctly.
Reposition to file:block 1:600
Block 10601 re-read correctly.
Reposition to file:block 1:9999
Block 20000 re-read correctly.
=== Test Succeeded. End Write, rewind, and re-read test ===



=== Append files test ===

This test is essential to Bacula.

I'm going to write one record  in file 0,
                  two records in file 1,
            and three records in file 2

btape: btape.c:578 Rewound "TS3200-1-Drive-0" (/dev/nst3)
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
```

```
btape: btape.c:608 Wrote 1 EOF to "TS3200-1-Drive-0" (/dev/nst3)
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:608 Wrote 1 EOF to "TS3200-1-Drive-0" (/dev/nst3)
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:608 Wrote 1 EOF to "TS3200-1-Drive-0" (/dev/nst3)
btape: btape.c:476 open device "TS3200-1-Drive-0" (/dev/nst3): OK
btape: btape.c:578 Rewound "TS3200-1-Drive-0" (/dev/nst3)
btape: btape.c:1418 Now moving to end of medium.
btape: btape.c:629 Moved to end of medium.
We should be in file 3. I am at file 3. This is correct!

Now the important part, I am going to attempt to append to the tape.

btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:608 Wrote 1 EOF to "TS3200-1-Drive-0" (/dev/nst3)
btape: btape.c:578 Rewound "TS3200-1-Drive-0" (/dev/nst3)
Done appending, there should be no I/O errors

Doing Bacula scan of blocks:
1 block of 64448 bytes in file 1
End of File mark.
2 blocks of 64448 bytes in file 2
End of File mark.
3 blocks of 64448 bytes in file 3
End of File mark.
1 block of 64448 bytes in file 4
End of File mark.
Total files=4, blocks=7, bytes = 451,136
End scanning the tape.
We should be in file 4. I am at file 4. This is correct!

The above Bacula scan should have output identical to what follows.
Please double check it ...
=== Sample correct output ===
1 block of 64448 bytes in file 1
End of File mark.
2 blocks of 64448 bytes in file 2
End of File mark.
3 blocks of 64448 bytes in file 3
End of File mark.
```

```
1 block of 64448 bytes in file 4
End of File mark.
Total files=4, blocks=7, bytes = 451,136
=== End sample correct output ===

If the above scan output is not identical to the
sample output, you MUST correct the problem
or Bacula will not be able to write multiple Jobs to
the tape.


=== Write, backup, and re-read test ===

I'm going to write three records and an EOF
then backup over the EOF and re-read the last record.
Bacula does this after writing the last block on the
tape to verify that the block was written correctly.

This is not an *essential* feature ...

btape: btape.c:578 Rewound "TS3200-1-Drive-0" (/dev/nst3)
btape: btape.c:813 Wrote first record of 64412 bytes.
btape: btape.c:824 Wrote second record of 64412 bytes.
btape: btape.c:835 Wrote third record of 64412 bytes.
btape: btape.c:608 Wrote 1 EOF to "TS3200-1-Drive-0" (/dev/nst3)
btape: btape.c:851 Backspaced over EOF OK.
btape: btape.c:856 Backspace record OK.
btape: btape.c:874
Block re-read correct. Test succeeded!
=== End Write, backup, and re-read test ===



=== Forward space files test ===

This test is essential to Bacula.

I'm going to write five files then test forward spacing

btape: btape.c:578 Rewound "TS3200-1-Drive-0" (/dev/nst3)
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:608 Wrote 1 EOF to "TS3200-1-Drive-0" (/dev/nst3)
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:608 Wrote 1 EOF to "TS3200-1-Drive-0" (/dev/nst3)
btape: btape.c:1905 Wrote one record of 64412 bytes.
```

```
btape: btape.c:1907 Wrote block to device.
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:608 Wrote 1 EOF to "TS3200-1-Drive-0" (/dev/nst3)
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:608 Wrote 1 EOF to "TS3200-1-Drive-0" (/dev/nst3)
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:608 Wrote 1 EOF to "TS3200-1-Drive-0" (/dev/nst3)
btape: btape.c:578 Rewound "TS3200-1-Drive-0" (/dev/nst3)
btape: btape.c:1632 Now forward spacing 1 file.
We should be in file 1. I am at file 1. This is correct!
btape: btape.c:1644 Now forward spacing 2 files.
We should be in file 3. I am at file 3. This is correct!
btape: btape.c:578 Rewound "TS3200-1-Drive-0" (/dev/nst3)
btape: btape.c:1657 Now forward spacing 4 files.
We should be in file 4. I am at file 4. This is correct!

btape: btape.c:1675 Now forward spacing 1 more file.
We should be in file 5. I am at file 5. This is correct!

=== End Forward space files test ===


Ah, I see you have an autochanger configured.
To test the autochanger you must have a blank tape
that I can write on in Slot 1.

Do you wish to continue with the Autochanger test? (y/n): y


=== Autochanger test ===

3301 Issuing autochanger "loaded" command.
Slot 1 loaded. I am going to unload it.
3302 Issuing autochanger "unload 1 0" command.
unload status=OK 0
3303 Issuing autochanger "load 1 0" command.
3303 Autochanger "load 1 0" status is OK.
btape: btape.c:476 open device "TS3200-1-Drive-0" (/dev/nst3): OK
btape: btape.c:1562 Rewound "TS3200-1-Drive-0" (/dev/nst3)
btape: btape.c:1569 Wrote EOF to "TS3200-1-Drive-0" (/dev/nst3)
```

```
The test autochanger worked!!
*quit
[root@atena bin]# ./btape -c ../etc/bacula-sd.conf /dev/nst2
Tape block granularity is 1024 bytes.
btape: butil.c:284 Using device: "/dev/nst2" for writing.
21-Nov 16:22 btape JobId 0: 3301 Issuing autochanger "loaded? drive 1" command.
21-Nov 16:22 btape JobId 0: 3302 Autochanger "loaded? drive 1", result is Slot 2.
btape: btape.c:476 open device "TS3200-1-Drive-1" (/dev/nst2): OK
*test

=== Write, rewind, and re-read test ===

I'm going to write 10000 records and an EOF
then write 10000 records and an EOF, then rewind,
and re-read the data to verify that it is correct.

This is an *essential* feature ...

btape: btape.c:1148 Wrote 10000 blocks of 64412 bytes.
btape: btape.c:608 Wrote 1 EOF to "TS3200-1-Drive-1" (/dev/nst2)
btape: btape.c:1164 Wrote 10000 blocks of 64412 bytes.
btape: btape.c:608 Wrote 1 EOF to "TS3200-1-Drive-1" (/dev/nst2)
btape: btape.c:1206 Rewind OK.
10000 blocks re-read correctly.
Got EOF on tape.
10000 blocks re-read correctly.
=== Test Succeeded. End Write, rewind, and re-read test ===

btape: btape.c:1274 Block position test
btape: btape.c:1286 Rewind OK.
Reposition to file:block 0:4
Block 5 re-read correctly.
Reposition to file:block 0:200
Block 201 re-read correctly.
Reposition to file:block 0:9999
Block 10000 re-read correctly.
Reposition to file:block 1:0
Block 10001 re-read correctly.
Reposition to file:block 1:600
Block 10601 re-read correctly.
Reposition to file:block 1:9999
Block 20000 re-read correctly.
=== Test Succeeded. End Write, rewind, and re-read test ===



=== Append files test ===

This test is essential to Bacula.
```

```
I'm going to write one record  in file 0,
                  two records in file 1,
            and three records in file 2

btape: btape.c:578 Rewound "TS3200-1-Drive-1" (/dev/nst2)
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:608 Wrote 1 EOF to "TS3200-1-Drive-1" (/dev/nst2)
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:608 Wrote 1 EOF to "TS3200-1-Drive-1" (/dev/nst2)
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:608 Wrote 1 EOF to "TS3200-1-Drive-1" (/dev/nst2)
btape: btape.c:476 open device "TS3200-1-Drive-1" (/dev/nst2): OK
btape: btape.c:578 Rewound "TS3200-1-Drive-1" (/dev/nst2)
btape: btape.c:1418 Now moving to end of medium.
btape: btape.c:629 Moved to end of medium.
We should be in file 3. I am at file 3. This is correct!

Now the important part, I am going to attempt to append to the tape.

btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:608 Wrote 1 EOF to "TS3200-1-Drive-1" (/dev/nst2)
btape: btape.c:578 Rewound "TS3200-1-Drive-1" (/dev/nst2)
Done appending, there should be no I/O errors

Doing Bacula scan of blocks:
1 block of 64448 bytes in file 1
End of File mark.
2 blocks of 64448 bytes in file 2
End of File mark.
3 blocks of 64448 bytes in file 3
End of File mark.
1 block of 64448 bytes in file 4
End of File mark.
Total files=4, blocks=7, bytes = 451,136
End scanning the tape.
We should be in file 4. I am at file 4. This is correct!

The above Bacula scan should have output identical to what follows.
```

```
Please double check it ...
=== Sample correct output ===
1 block of 64448 bytes in file 1
End of File mark.
2 blocks of 64448 bytes in file 2
End of File mark.
3 blocks of 64448 bytes in file 3
End of File mark.
1 block of 64448 bytes in file 4
End of File mark.
Total files=4, blocks=7, bytes = 451,136
=== End sample correct output ===

If the above scan output is not identical to the
sample output, you MUST correct the problem
or Bacula will not be able to write multiple Jobs to
the tape.


=== Write, backup, and re-read test ===

I'm going to write three records and an EOF
then backup over the EOF and re-read the last record.
Bacula does this after writing the last block on the
tape to verify that the block was written correctly.

This is not an *essential* feature ...

btape: btape.c:578 Rewound "TS3200-1-Drive-1" (/dev/nst2)
btape: btape.c:813 Wrote first record of 64412 bytes.
btape: btape.c:824 Wrote second record of 64412 bytes.
btape: btape.c:835 Wrote third record of 64412 bytes.
btape: btape.c:608 Wrote 1 EOF to "TS3200-1-Drive-1" (/dev/nst2)
btape: btape.c:851 Backspaced over EOF OK.
btape: btape.c:856 Backspace record OK.
btape: btape.c:874
Block re-read correct. Test succeeded!
=== End Write, backup, and re-read test ===



=== Forward space files test ===

This test is essential to Bacula.

I'm going to write five files then test forward spacing

btape: btape.c:578 Rewound "TS3200-1-Drive-1" (/dev/nst2)
btape: btape.c:1905 Wrote one record of 64412 bytes.
```

```
btape: btape.c:1907 Wrote block to device.
btape: btape.c:608 Wrote 1 EOF to "TS3200-1-Drive-1" (/dev/nst2)
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:608 Wrote 1 EOF to "TS3200-1-Drive-1" (/dev/nst2)
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:608 Wrote 1 EOF to "TS3200-1-Drive-1" (/dev/nst2)
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:608 Wrote 1 EOF to "TS3200-1-Drive-1" (/dev/nst2)
btape: btape.c:1905 Wrote one record of 64412 bytes.
btape: btape.c:1907 Wrote block to device.
btape: btape.c:608 Wrote 1 EOF to "TS3200-1-Drive-1" (/dev/nst2)
btape: btape.c:578 Rewound "TS3200-1-Drive-1" (/dev/nst2)
btape: btape.c:1632 Now forward spacing 1 file.
We should be in file 1. I am at file 1. This is correct!
btape: btape.c:1644 Now forward spacing 2 files.
We should be in file 3. I am at file 3. This is correct!
btape: btape.c:578 Rewound "TS3200-1-Drive-1" (/dev/nst2)
btape: btape.c:1657 Now forward spacing 4 files.
We should be in file 4. I am at file 4. This is correct!

btape: btape.c:1675 Now forward spacing 1 more file.
We should be in file 5. I am at file 5. This is correct!

=== End Forward space files test ===


Ah, I see you have an autochanger configured.
To test the autochanger you must have a blank tape
that I can write on in Slot 1.

Do you wish to continue with the Autochanger test? (y/n): y


=== Autochanger test ===

3301 Issuing autochanger "loaded" command.
Slot 2 loaded. I am going to unload it.
3302 Issuing autochanger "unload 2 1" command.
```

```
unload status=OK 0
3303 Issuing autochanger "load 1 1" command.
3303 Autochanger "load 1 1" status is OK.
btape: btape.c:476 open device "TS3200-1-Drive-1" (/dev/nst2): OK
btape: btape.c:1562 Rewound "TS3200-1-Drive-1" (/dev/nst2)
btape: btape.c:1569 Wrote EOF to "TS3200-1-Drive-1" (/dev/nst2)

The test autochanger worked!!
```

Apêndice B:
Testes de velocidade e de preenchimento de fitas com o utilitário btape

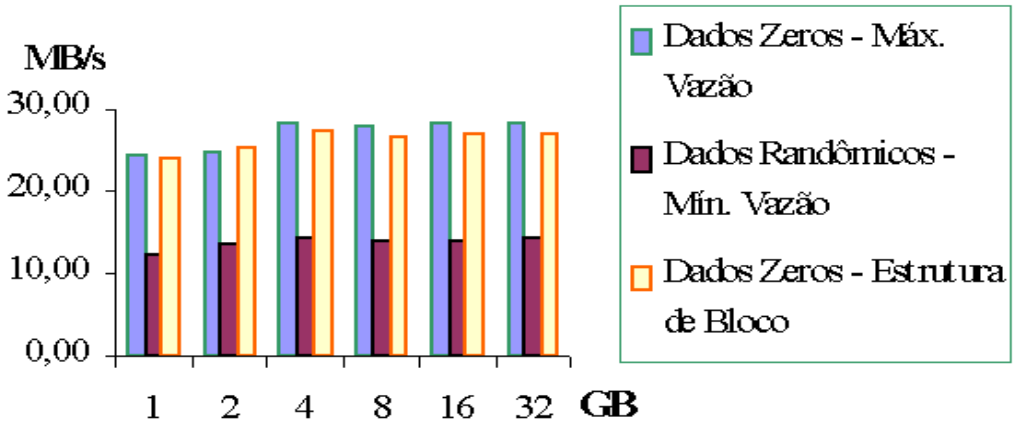


Figura 29. Teste de Velocidade (LTO-1)

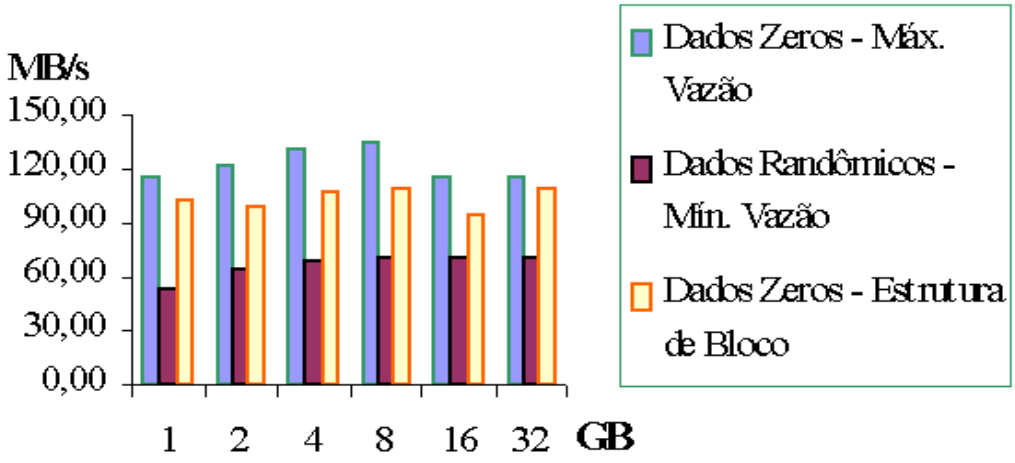


Figura 30. Teste de Velocidade (LTO-3)

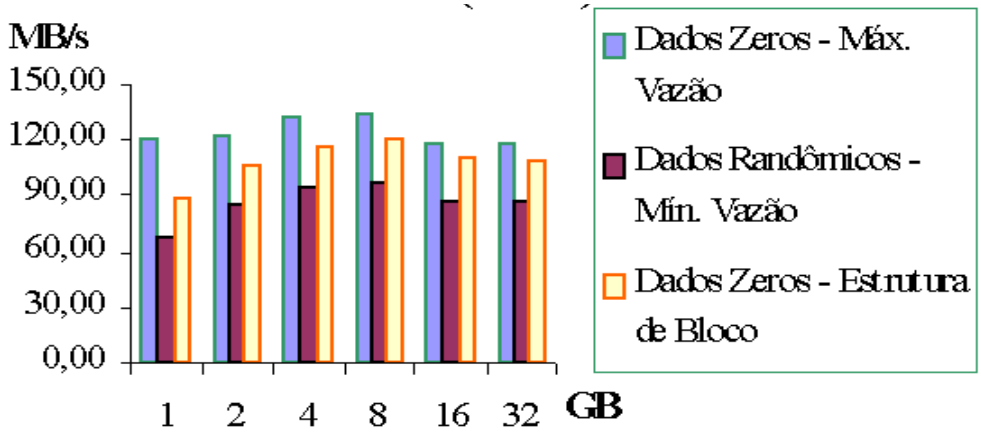


Figura 31. Teste de Velocidade (LTO-4)

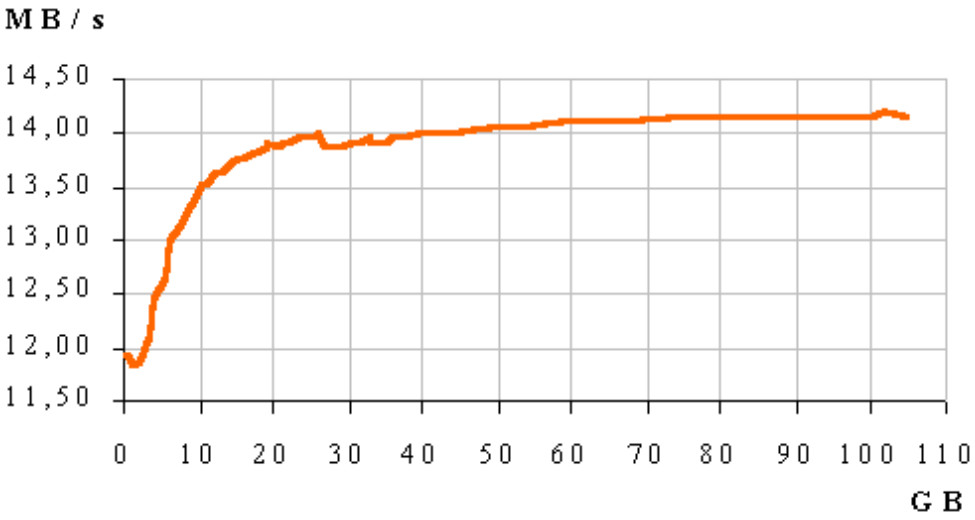


Figura 32. Velocidade de Preenchimento da Fita LTO-1.

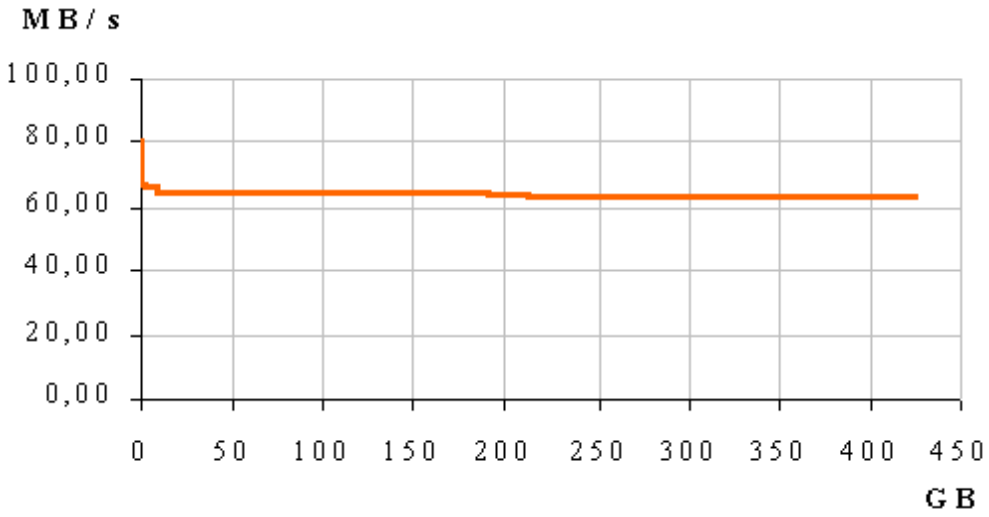


Figura 33. Velocidade de Preenchimento da Fita LTO-3.

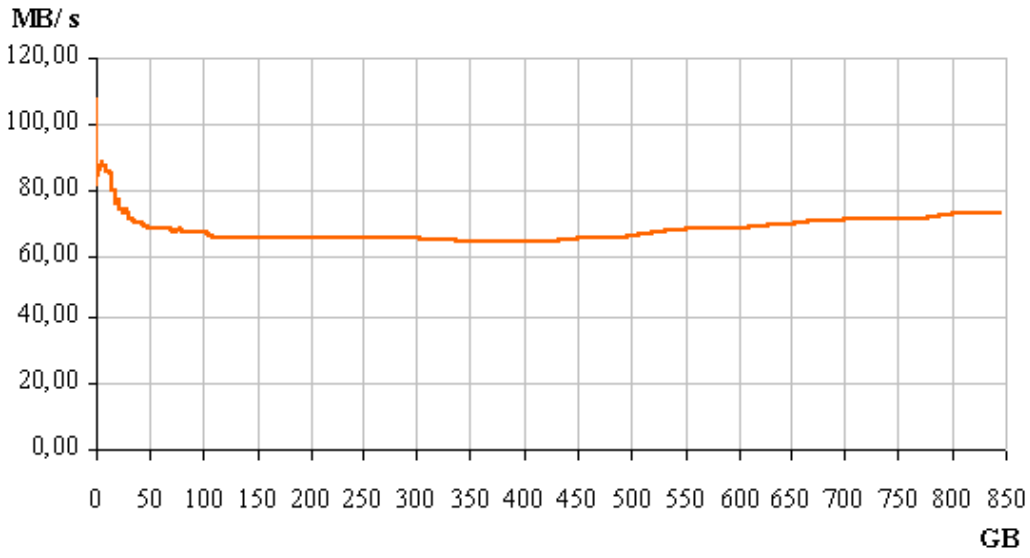


Figura 34. Velocidade de Preenchimento da Fita LTO-4.

Anexo A:
Software Livre

Software open source é o *software* de código fonte aberto, ou seja, é disponibilizado gratuitamente em código fonte e/ou compilado, permitindo, através de sua licença, utilização livre, alterações e distribuições de seus derivados sob a mesma licença sem ônus. Com essa liberdade é possível melhor compreensão e customização do *software* às necessidades da empresa, além da grande economia em investimentos de TI. Em contra partida aos softwares proprietários, que, na maioria das vezes, cobram por suas licenças valores muito altos com renovação a cada ano.

Um *software Open Source* (Código Aberto) é uma nomenclatura de certificação usada pela *Open Source Initiative* (OSI) para apresentar o *Software* Livre, termo este usado pela *Free Software Foundation* (FSF), à empresas de uma forma mais comercial, puramente técnica, evitando os aspectos éticos, direitos e liberdades em que se baseia o *software* livre.

Qualquer licença de *software* livre é também uma licença de código aberto. Genericamente, trata-se de *software* que respeita as quatro liberdades definidas pela FSF.

1ª liberdade:

A liberdade de executar o *software*, para qualquer uso.

2ª liberdade:

A liberdade de estudar o funcionamento de um programa e de adaptá-lo às suas necessidades.

3ª liberdade:

A liberdade de redistribuir cópias.

4ª liberdade:

A liberdade de melhorar o programa e de tornar as modificações públicas de modo que a comunidade inteira beneficie da melhoria.

Além destes direitos a licença também impõe os seguintes deveres:

- publicar em cada cópia um aviso de direitos autorais (*copyright*) e uma notificação sobre a ausência de garantia;

- redistribuir as alterações porventura realizadas juntamente com uma cópia da licença;

- distribuir as alterações incluindo o código-fonte correspondente completo.

Algumas grandes empresas como IBM, HP, Intel e Dell também têm investido no *software* de código aberto, juntando esforços para a criação do *Open Source Development Lab* (OSDL), instituição destinada à criação de tecnologias de código aberto.

Anexo B:
Taxonomia das Licenças de *Software*

Tabela 8. Taxonomia das licenças de *software*.

Software Type							
Commercial							
Trial Software	***X*** *(Non-full featured)*	***X***					
Non-Commercial Use	***X*** *(Usage dependent)*	***X***					
Shareware	***X*** *(Unenforced licensing)*	***X***					
Royalty-free binaries *("Freeware")*	***X***	***X***	***X***				
Royalty-free libraries	***X***	***X***	***X***	***X***			
Open Source *(BSD-Style)*	***X***	***X***	***X***	***X***	***X***		
Open Source *(Apache Style)*	***X***	***X***	***X***	***X***	***X***	***X***	
Open Source *(Linux/GNU style)*	***X***	***X***	***X***	***X***	***X***	***X***	***X***
License Feature	***Zero Price Avenue***	***Redistributable***	***Unlimited Usage***	***Source Code Available***	***Source Code Modifiable***	***Public "Check-ins" to core codebase***	***All derivatives must be free***

Fonte: http://www.catb.org/~esr/halloween/index.html