FACULDADES INTEGRADAS DO BRASIL - UNIBRASIL
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

LEONARDO ANTÔNIO DOS SANTOS

COMPARTILHAMENTO DE ARQUIVOS EM STORAGE DE ALTA DISPONIBILIDADE NA UNIBRASIL

TRABALHO DE CONCLUSÃO DE CURSO I

CURITIBA

2012

LEONARDO ANTÔNIO DOS SANTOS

COMPARTILHAMENTO DE ARQUIVOS EM STORAGE DE ALTA DISPONIBILIDADE NA UNIBRASIL

Trabalho de Conclusão de Curso I apresentado ao Curso de Bacharelado em Sistemas de Informação da Faculdades Integradas do Brasil - Unibrasil.

Orientador: Msc. Sabrina Victório

Co-orientador: Msc. Pedro Eugênio Rocha

CURITIBA

2012

LISTA DE FIGURAS

LISTA DE SIGLAS

HDs	*Hard Disk* - Disco Rígido
NFS	*Network File System*
SMB	*Server Message Block*
FC	*Fibre Channel*
AoE	*ATA Over Ethernet*
ATA	*Advanced Technology Attachment*

LISTA DE ACRÔNIMOS

CIFS	*Common Internet File System*
iSCSI	*Internet Small Computer System Interface*
SAN	*Storage Area Network*
NAS	*Network Attached Storage*
LAN	*Local Area Network*
RAID	*Redundant Array of Independent Disks*

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo será apresentada uma visão geral sobre o estado da arte, que será melhor detalhado no Capítulo 2. Na Seção 1.1 será feita uma explanação dos problemas enfrentados pela Unibrasil no armazenamento e disponibilização de dados que motivaram a condução deste trabalho. A Seção 1.2 e 1.3 trata dos objetivos gerais e específicos daquilo que se busca após a conclusão deste trabalho. A Seção 1.4 apresenta os principais motivos que justificam os resultados esperados diante dos problemas expostos na Seção 1.1.

De modo geral, as tecnologias têm se tornado cada vez mais essenciais ao dia a dia das empresas e das pessoas. Grandes corporações têm utilizado sistemas informatizados para gerenciar as suas linhas de produção, sistemas de controle de estoque e compartilhamento de arquivos. Especialmente sobre o compartilhamento de arquivos, os volumes de dados têm aumentado em proporção cada vez maior. Basta observar o crescimento de serviços de armazenamento de dados famosos como *YouTube* (YOUTUBE..., 2011), *4shared* (4SHARED..., 2005) e *Source Forge* (SOURCEFORGE..., 2011). Além da necessidade de compartilhamento de grandes quantidades de dados, grande parte das empresas possuem informações que precisam ser armazenados de forma segura e com alta disponibilidade, ou seja, a informação necessita estar disponível quase todo tempo, senão todo o tempo.

Assim, é fácil pensar em juntar vários HDs (*Hard Disk* - Disco Rígido) a fim de solucionar o problema do compartilhamento de arquivos. Mas esta solução, apesar de parecer fácil, na prática não funciona lançando mão para isto de métodos de compartilhamento de arquivos tradicionais, como CIFS, NFS ou SMB — protocolos para compartilhamento de arquivos para sistemas de arquivos (*filesystems*) como *Windows* ou *Linux*. A justificativa para isto é porque estes métodos compartilham apenas os arquivos e não os dispositivos de armazenamento, tornando inviável o gerenciamento deste dispositivo de forma remota.

Neste contexto, surgiram outros protocolos para compartilhamento de dispositivos de armazenamento de dados (ou *block devices*), sendo os principais deles o iSCSI (*Internet Small Computer System Interface*) (SATRAN et al., 2004) e o FC (*Fibre Channel*) (CLARK, 1999).

Ambos os protocolos cumprem os objetivos do armazenamento de dados de forma segura, com alta disponibilidade e para grandes quantidades de arquivos. Os principais problemas que envolvem estes protocolos são o alto custo de tecnologia e a dependência de hardware específico para implementação.

O AoE (*ATA Over Ethernet*) é um protocolo *open source* (código aberto) desenvolvido pela empresa *Coraid* (HOPKINS; COILE, 2001) e disponibilizado à comunidade de ciências da computação a fim de ser estudado e melhorado em sua implementação. Ele resolve os problemas mais comuns do *Fibre Channel* e iSCSI que são, respectivamente, alto custo e sobrecarga de rede.

1.1 PROBLEMA

Antigamente, quando um professor desejava compartilhar um arquivo, seja texto, áudio ou vídeo, gravava o seu conteúdo em uma mídia (discos removíveis, CD-ROMs, DVD-ROMs ou *flash-drives*) e repassava uma cópia entre todos os alunos ou distribuía várias cópias entre eles. Com o passar do tempo, o acesso à intranet e internet assim como a computadores pessoais, *notebooks*, *tablets* e *smartphones* se tornou mais acessível a todos. Porém, as tecnologias que proviam o acesso aos dados não evoluíram na mesma proporção. Deste modo, as instituições de ensino se viram diante da falta de uma área de armazenamento de dados compartilhada entre todo o corpo docente e discente. Na UniBrasil esta realidade não é diferente.

Outro grande problema é o alto custo nas tecnologias de armazenamento de dados para compartilhamento de arquivos alta disponibilidade. Não basta compartilhar dados, é necessário que estes dados estejam sempre disponíveis quando solicitados e livres de desastre casos de perda de algum dos discos. Para solucionar este problema, o custo de tecnologias como Fibre Channel e iSCSI nativo é alto em relação a tecnologias como AoE, pois exigem hardware específico para este fim (ZHOU; CHUANG, 2009).

Por fim, todos os dias uma grande quantidade de hardware sem uso é lançado ao meio ambiente, seja em grandes salas preparadas para recebê-los ou em locais não tão apropriados para este fim, como lixões. Implementações de protocolos e ferramentas em software surgiram a fim de conectar vários discos modernos a fim de aumentar a sua performance em conjunto. Recentemente surgiu um protocolo e as ferramentas de seu uso em código livre que torna possível a reutilização de hardwares comuns através da sua integração, dando a eles o comportamento de hardwares mais modernos e trazendo uma série de benefícios para a sociedade, principalmente através do cuidado com o meio ambiente.

1.2 OBJETIVO GERAL

O objetivo geral deste trabalho é desenhar e juntar tecnologias da área de armazenamento de dados dentro de uma arquitetura que, com a utilização do protocolo de compartilhamento de discos em software livre AoE juntamente com outras ferramentas de software livre, disponibilize ao corpo docente e discente da UniBrasil uma área de armazenamento de dados com altos padrões de disponibilidade e desempenho para ser usada em conjunto com aplicações de compartilhamento de arquivos mais diversos, permitindo aos usuários compartilharem os seus arquivos.

A estrutura criada pelo trabalho pode ser utilizada para compartilhar dados principalmente entre alunos do curso de Sistemas de Informação, porém, de acordo com que implementações de serviços e outros trabalhos desenvolvidos no curso forem utilizando os recursos oferecidos pelo trabalho em questão, informações poderão também ser compartilhadas entre alunos de toda a instituição. Com a homologação da arquitetura, esta poderá ser publicada em portais de software livre para que possa resolver problemas de armazenamento a qualquer que assim desejar.

1.3 OBJETIVOS ESPECÍFICOS

- Estudar sobre protocolos de rede, alta disponibilidade de dados, analise de performance e aplicações web;

- Avaliar tecnologias paralelas que já encontram-se em uso no mercado;

- Implementar o uso do protocolo adaptando-o a uma infraestrutura capaz de compartilhar dados pela intranet e pela internet via web de forma distribuída;

- Desenvolver sistema web que facilite o uso desta tecnologia ao usuário final.

1.4 JUSTIFICATIVA

O compartilhamento de arquivos, apesar de ser simples utilizando as ferramentas conhecidas, torna-se muito custoso quando se busca implementar grandes áreas de armazenamento de dados, principalmente se forem distribuídos e com alta disponibilidade. Isto acontece devido ao fato desta implementação ser possível somente com a compra de equipamentos especializados para este fim.

O protocolo AoE aparece com a finalidade de resolver problemas de compartilhamento

de arquivos, só que a um baixo custo de tecnologia, se comparado a protocolos para *filesystems* (sistemas de arquivos) distribuídos.

Uma implementação completa deste protocolo juntamente com uma ferramenta Web que facilitará e orquestrará o seu uso.

2 REVISÃO DE LITERATURA

Neste capítulo será apresentada em detalhes a principal tecnologia em que este trabalho se baseia, o protocolo AoE (*ATA Over Ethernet*) assim como tecnologias similares e a comparação entre elas, justificando o não uso destas similares neste trabalho.

2.1 COMPARTILHAMENTOS

Através do que se tem na literatura, SAN (*Storage Area Network*) e NAS (*Network Attached Storage*) são termos potencialmente fáceis de se confundir. NAS significa *compartilhar arquivos* ao passo de que SAN significa *compartilhar discos*. A principal diferença entre eles está em que o no NAS o cliente acessa arquivos e diretórios individuais compartilhados pelo servidor, no SAN o cliente tem acesso ao dispositivo físico real podendo realizar qualquer tipo de operação sobre ele (FUNDAMENTALS. . . , 2008).

2.1.1 COMPARTILHAMENTO DE ARQUIVOS (NAS)

O compartilhamento de arquivos é uma das formas mais comuns de *storages de rede* e comumente usado nos sistemas operacionais versões *home* para Windows e Macintoch. Um computador em particular (servidor de arquivos) permite outros computadores acessarem alguns dos seus arquivos e/ou diretórios.

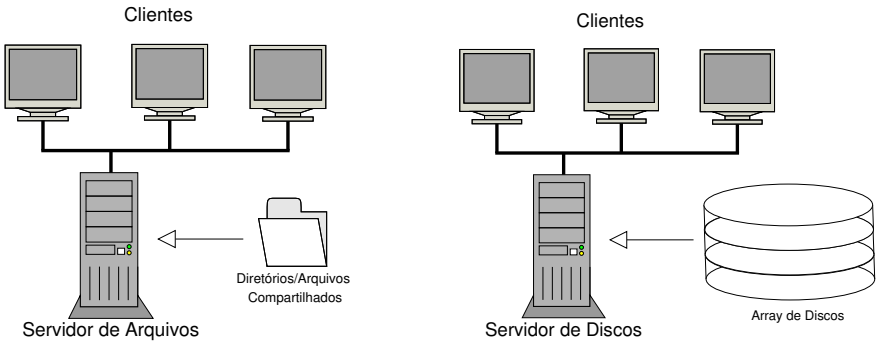
Para compartilhar seus arquivos, um protocolo de compartilhamento de arquivos é utilizado. Os protocolos mais conhecidos nesta categoria são NFS, CIFS (formalmente chamado de SMB, a base do compartilhamento Windows e implementado no Linux pelo protocolo SAMBA), FTP, HTTP entre outros. A Figura (a) mostra uma típica conexão de clientes a um servidor de arquivos através de uma LAN (*Local Area Network*, ou rede local) (FUNDAMENTALS. . . , 2008).

Em uma utilização mais realista, como por exemplo um ambiente corporativo, um servidor de arquivos normalmente possui muitos discos configurados em RAID (*Redundant*

Array of Independent Disks — conjunto de discos redundantes), que será tratado na Seção 2.3.1. O servidor é formatado em um sistema de arquivos (EXT[2-4], NTFS, entre outros) e mantém um conjunto de arquivos e diretórios compartilhados com os clientes.

No cenário dos compartilhamentos de arquivos NAS, uma mesma máquina pode ser cliente e servidor, pois em ambientes pequenos é comum todas as máquinas compartilharem um ou mais de seus arquivos com todas as outras. Já em um ambiente corporativo, é comum clientes Windows acessarem via CIFS/SMB um servidor que conecta a vários dispositivos de armazenamento maiores via NFS. Além disso, é bastante utilizado o empilhamento de compartilhamentos de arquivos através da rede, de modo que às vezes um servidor chega a fazer até o armazenamento local através da rede.

Por fim, o compartilhamento de arquivos pode ser um gargalo de desempenho, uma vez que um único servidor poderá gerenciar muitos tipos de operações sobre arquivos (leitura, escrita, criação, exclusão, incremento, etc.). Quando um cliente escreve um arquivo no servidor, na verdade o arquivo é escrito duas vezes, uma no cliente no seu *virtual shared file space* (espaço de arquivos virtual compartilhado) e outra no servidor, no disco real (LANDOWSKI; CURRAN, 2011).



(a) Arquitetura padrão NAS (b) Arquitetura padrão SAN

Figura 1: Servidor NAS e SAN.

2.1.2 COMPARTILHAMENTO DE DISCOS (SAN)

O compartilhamento de discos é mais simples que o de arquivos, porém é bem menos conhecido pela maioria das pessoas. Nem sempre o termo *"compartilhar"* possui o sentido de compartilhamento arquivos. Normalmente um servidor de discos compartilha um volume de disco (disco real ou virtual, caso se utilize LVM (LEWIS, 2006) ou similar) com apenas um computador por vez. O computador monta o volume e o usa (o termo monta remete às montagens de fitas citadas nos manuais de fitas de 1960). Ao compartilhamento de discos por muitas máquinas pode acontecer quando o disco for formatado com um *cluster filesystem*

(VMWare VMFS (VAGHANI, 2010), OCFS (FASHEH, 2006), etc.) entre outros. Este tipo de abordagem é utilizada para aplicações em que é necessário que múltiplos *hosts* acessem o disco compartilhado, como é o caso da virtualização em *cluster* (FUNDAMENTALS. . . , 2008).

2.2 PROTOCOLOS PARA COMPARTILHAMENTO DE DISCOS

2.2.1 ATA OVER ETHERNET (AOE)

De forma sumária, o AoE é um protocolo de padrão aberto que permite o acesso direto de *hosts* (clientes de rede) a dispositivos de disco, realizando isto através de uma rede. Com ele é possível compartilhar discos simples ou *arrays* de disco (conjuntos de discos) usufruindo de todas as vantagens de acesso *raw device* (acesso direto ao dispositivo) e ser *layer 2* (camada *Ethernet*) (HOPKINS; COILE, 2001).

O protocolo AoE foi construído para ser simples, de alta performance e baixo custo em tecnologia, se comparado às principais tecnologias existentes no mercado para o mesmo fim, como iSCSI e *Fibre Channel*, quando o objetivo é compartilhar *block device* (dispositivo de bloco), pois aposta na principal vantagem da simplicidade eliminando o *overhead* TCP/IP.

ATA (*Advanced Technology Attachment*) é um conjunto padronizado de comandos para os dispositivos de armazenamento mais comuns. O AoE encapsula os comandos de um cliente ATA através do protocolo *Ethernet* e repassa ao servidor e vice-versa. O cliente AoE apenas solicita ao servidor qual *disk block* (bloco de disco), em qual disco e qual *shelf* (gaveta - ou conjunto de discos) em que está a sua informação e este bloco é novamente encapsulado pelo enlace de dados no sentido servidor-cliente.

Estas vantagens permitem ao AoE não simplesmente acessar os arquivos de um disco remoto, mas realizar operações de baixo nível em dispositivos de disco, como formatar sistemas de arquivos e recriar a tabela de partições. Por padrão o AoE é nativo nos *kernels* (núcleo do sistema) acima da versão 2.6.11.

2.2.2 ISCSI

O Internet SCSI, ou iSCSI, é um dos mais conhecidos protocolos para Storgaes SAN. Semelhante ao AoE, o iSCSI também encapsula comandos, mas, no seu caso são comandos SCSI. Isto também permite ao iSCSI a qualidade de storage SAN implementado sem o uso de componentes caros. Por exemplo, uma interface *gigabit Ethernet* custa entre R$ 70,00 e R$ 250,00 (AIKEN et al., 2003).

SCSI é uma popular família de protocolos que torna possível sistemas se comunicarem com dispositivos E/S (entrada e saída), especialmente dispositivos de armazenamento. Estes protocolos nada mais são do que protocolos de aplicação *request/response* com um modelo de arquitetura comum padronizado e um conjunto de comandos, bem como um padronizado conjunto de comandos para diferentes classes de dispositivos (discos, fitas, pendrives etc.) (SATRAN et al., 2004).

Comparativamente, o tanto o ATA quanto o SCSI cumprem a mesma finalidade, ser um conjunto padrão de comandos para conectividade com dispositivos de armazenamento em geral. Tradicionalmente, o ATA era considerado mais barato e simples e o SCSI mais caro e robusto, mas esta comparação já não é mais verdadeira. Ambos agora possuem DMA (*direct memory access* - acesso direto à memória), que elimina o problema de interrupções à CPU durante o processo de leitura ou escrita. Ambos também podem fazer enfileiramento de comandos fora de ordem para a CPU. E ambos possuem recurso de *hotswap*, que permite o dispositivo ser removido e conectado sem problemas com o sistema em execução (WHAT..., 2006).

Tanto ATA quanto SCSI foram criados para arquiteturas de transferência de dados originalmente paralela, mas atualmente já se utilizam estes padrões em arquiteturas serial com o SAS (Serial Attached SCSI) e SATA (Serial ATA). Quanto à velocidade, é difícil dizer qual padrão é mais rápido do que o outro, pois isto seria como comparar processadores diferentes para julgar o desempenho de um sistema. A velocidade não depende apenas do conector, mas da quantidade de giros do disco, o quão rápido a cabeça de leitura se move entre outros fatores (WHAT..., 2006).

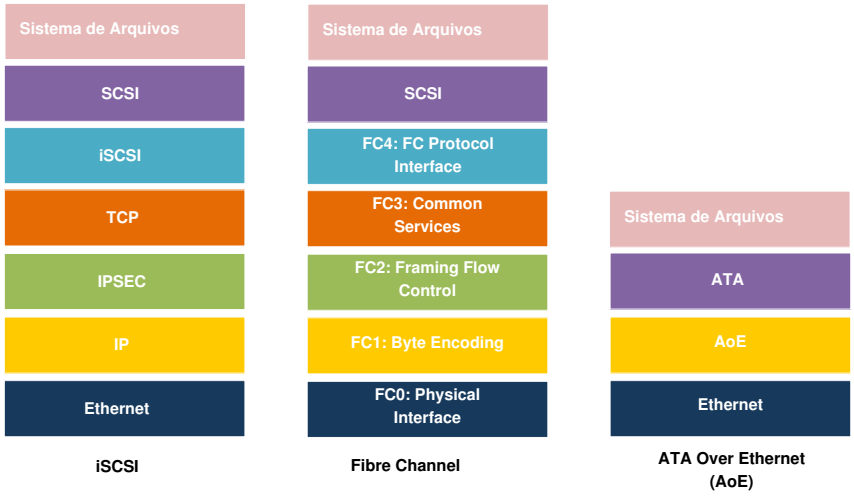


Figura 2: Comparativo entre protocolos SAN existentes.

2.3 DESEMPENHO E ALTA DISPONIBILIDADE

2.3.1 RAID

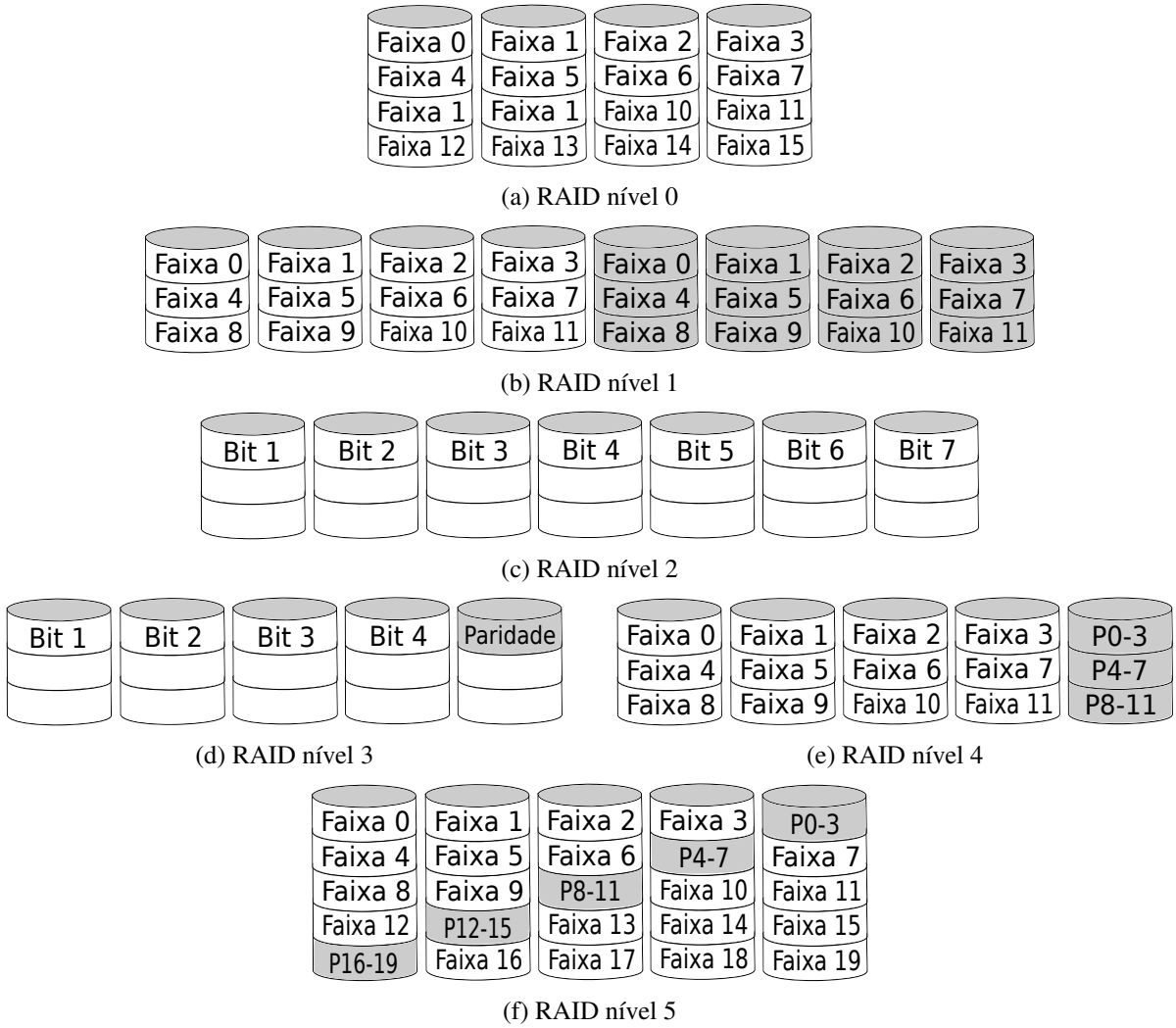
Nos últimos anos, o desempenho das CPUs tem crescido de forma exponencial, chegando a duplicar a cada 18 meses, o que não acontece com os dispositivos de disco. De 1970 aos dias atuais, o tempo médio de movimentação da cabeça de leitura (*seek time*) dos discos melhorou de 50 a 100 ms para menos de 10 ms. Para outros ramos da tecnologia este avanço poderia ser bom, mas para a computação é um ganho muito pequeno. Assim, a disparidade entre CPUs e dispositivos de disco tem ficado cada vez mais acentuada (TANENBAUM, 2010).

Pelo fato do processamento paralelo tornar as CPUs mais rápidas, este fato levou muitos a acreditarem que o mesmo poderia ser feito para dispositivos de E/S a fim de agilizar o acesso a disco e diminuir a disparidade de desempenho entre os dispositivos de entrada e saída e os processadores assim como de confiabilidade. Neste sentido Patterson *et. al* sugeriram seis organizações para os discos e as definiu como RAID (*redundant array of inexpensive disks* - arranjo redundante de discos baratos), que a indústria adotou rapidamente e substituiu o *'I'* por 'Independentes', por questões mercadológicas (PATTERSON et al., 1988; TANENBAUM, 2010).

A ideia básica em que o RAID se baseia é juntar vários discos em uma única caixa e se apresentar ao sistema como um único disco, substituindo a placa controladora por uma placa RAID, visto que discos ATA e SCSI têm um bom desempenho, confiabilidade e baixo custo, podendo ainda permitir vários dispositivos em uma única controladora (15 para dispositivos mais robustos) (LOBUE et al., 2002).

O *RAID nível 0* é mostrado na Figura (a). Ele descreve um modelo de RAID baseado em *stripping*, que consiste na divisão dos dados em faixas, cada uma contendo k setores. A faixa 0 consiste nos setores de 0 a $k-1$, a faixa 1 consiste nos setores de k a $2k-1$ e assim por diante. O processo de gravação é realizado através de uma alternância consecutiva entre as faixas, mais conhecido como *round-robin*. No exemplo da Figura (a), quando uma operação de leitura é recebida, o controlador RAID quebra esta operação em outras quatro que são enviadas para as controladoras de disco e processadas em paralelo, aumentando o desempenho proporcionalmente ao número de discos que processam as requisições paralelas. Deste modo, o RAID0 trabalha melhor em casos de requisições maiores. Uma desvantagem do RAID0 acontece no caso da perda de dados, pois, como os dados estão divididos em várias faixas, a perda de um disco causa a perda total dos dados (TANENBAUM, 2010).

Outro tipo de RAID é o *RAID nível 1*, mostrado na Figura (b). Neste tipo de RAID os



(a) RAID nível 0

(b) RAID nível 1

(c) RAID nível 2

(d) RAID nível 3

(e) RAID nível 4

(f) RAID nível 5

Figura 3: RAID níveis 0 a 5. Os discos cópia de segurança e paridade estão sombreados. Fonte: (TANENBAUM, 2010)

todos os dados são duplicados em outros discos de forma que existam quatro discos primários e quatro discos de cópia de segurança (*backup*). Durante o processo de escrita, cada faixa é escrita duas vezes; durante o processo de leitura, os dados podem ser lidos de qualquer uma das faixas. Em consequência disto, o desempenho da escrita é pior do que o uso de um único disco, porém, o desempenho da leitura pode ser até duas vezes melhor, visto que pode se obter o dado de até dois discos em paralelo. A tolerância a falhas é a grande vantagem desta abordagem, considerando que a perda de um dos discos não ocasiona a perda dos dados, pois o disco de cópia passa a atuar no lugar do disco danificado. A restauração dos dados é feita apenas com a instalação de um novo disco, copiando para ele os dados da cópia de segurança.

Os *níveis 2 e 3 de RAID* não serão utilizados no desenvolvimento deste, mas merecem uma breve e

3 TRABALHOS CORRELATOS

4 METODOLOGIA DA PESQUISA

4.1 PARTE I - NATUREZA DA PESQUISA

Redes de Computadores e Sistemas Distribuídos (Protocolos de Comunicação, Arquitetura de *Clusters* e Servidores, Sistemas Distribuídos, Alta Disponibilidade).

4.2 PARTE II - TIPO DE PESQUISA

Aplicada e Bibliográfica.

4.3 PARTE III - LEVANTAMENTO DE REQUISITOS

Levantamento das necessidades de compartilhamento de arquivos utilizando-se de mecanismos como entrevistas e análise dos sistemas existentes.

4.4 DESCRIÇÃO E JUSTIFICATIVA DAS TECNOLOGIAS APLICADAS

As principais tecnologias utilizadas, desde ferramentas, linguagens de programação e protocolos serão melhores descritas nas próximas Subseções.

4.4.1 AOE TOOLS / VBLADE

O protocolo AoE (Ata over Ethernet) é utilizado na comunicação entre os dispositivos de armazenamento de dados, ou discos rígidos. Como este é um dos núcleos principal deste trabalho, foi melhor explanado na Seção 2.2.1. O AoE Tools é um conjunto de ferramentas gerenciamento do protocolo AoE, possibilitando executar comandos e manipular discos na *máquina initiator*. Para a *máquina target* é necessário a utilização do conjunto de ferramentas *vBlade* (*Virtual EtherDrive* ® *blade AoE target*) (ATA. . . , 2012).

Todas as particularidades citadas acima e visto a discussão encontrada na Seção 4.6 tornam indispensáveis o uso destes dois conjuntos de ferramentas na implementação deste trabalho.

4.4.2 PHP

O PHP (*Hypertext Preprocessor*) (PHP..., 2001) é uma das linguagens de programação mais utilizadas do mundo, especialmente para o desenvolvimento web. Um dos principais objetivos do PHP é possibilitar a implantação de soluções web rápidas, simples e eficientes, o que já tem conseguido há anos. Dentre as suas principais características técnicas estão: ser estruturado e orientado a objetos, possuir portabilidade (independência de plataforma), ter tipagem dinâmica (não precisa declarar tipo), possuir sintaxe similar ao C/C++ e Perl, e, principalmente, ser open-source.

Para o trabalho apresentado, o PHP é importante por conta do seu requisito dentro do *framework* de desenvolvimento utilizado, o CakePHP (CAKEPHP..., 2005) (descrito na Subseção 4.4.3), além das características já citadas acima.

4.4.3 CAKEPHP

O CakePHP Development Framework é um *framework* de desenvolvimento ágil para PHP que fornece uma arquitetura extensível para desenvolvimento, manutenção e implantação de aplicativos. Usando padrões de projeto conhecido como MVC (*Model-view-controller* - Modelo-Visão-Controlador) (TRYGVE/MVC, 2011) e ORM (*Object-relational mapping* - Mapeamento objeto-relacional) (AMBLER, 2002) no âmbito da convenção sobre o paradigma de configuração, o CakePHP reduz custos de desenvolvimento e ajuda os desenvolvedores a escreverem menos código (CAKEPHP..., 2005).

Dentre os motivos que levaram à escolha do CakePHP para o desenvolvimento deste trabalho, podemos dizer que o CakePHP agiliza o desenvolvimento com alto padrão de qualidade de software e, visto que a abordagem deste trabalho não é tipica de desenvolvimento, agilizando a implementação, ajuda a que pontos mais importantes do trabalho a não serem onerados. Um outro motivo é que o CakePHP tem total compatibilidade de ORM com os principais bancos de dados do mercado, como o MySQL, que será melhor detalhando na Subseção 4.4.4.

4.4.4 MYSQL

O MySQL é um SGBD (Sistema de Gerenciamento de Banco de Dados), que utiliza a linguagem SQL (Linguagem de Consulta Estruturada, do inglês *Structured Query Language*) como interface. É atualmente um dos bancos de dados mais populares, com mais de 10 milhões de instalações pelo mundo (WHY. . . , 2012b).

Em grande parte, o sucesso do MySQL deve-se à integração fácil com o PHP e estar incluído nos pacotes de hospedagem de sites da Internet oferecidos atualmente quase que de forma obrigatória. Grandes empresas como Yahoo! Finance, Motorola, NASA e Silicon Graphics utilizam o MySQL em suas aplicações de missão crítica (CASE. . . , 2012).

Além dos motivos já citados acima, o MySQL é de livre utilização, fácil uso e com vasta documentação publicada na Internet, o que torna relevante a sua utilização na implementação deste trabalho.

4.4.5 LINUX DEBIAN

O Linux Debian, ou simplesmente Debian, é o sistema operacional que será utilizado tanto nas *máquinas initiator* quando nas *máquinas target* uma vez que o Debian é uma distribuição não comercial livre (gratuita e de código fonte aberto) do GNU/Linux (amplamente utilizada). O Debian tornou-se bastante conhecido por causa do seu sistema de gestão de pacotes, chamado APT, que permite: atualizações relativamente fáceis a partir de versões realmente antigas; instalações quase sem esforço de novos pacotes; remoções limpas dos pacotes antigos e atualizações mais seguras dos pacotes nas máquinas, visto que os pacotes são baixados do repositório oficial (**??**).

Atualmente uma versão estável do Debian versão 6.0, codinome *"Squeeze"*, a qual será a versão utilizada neste trabalho. O Debian Stable procura sempre manter os pacotes mais estáveis, ou seja, alguns pacotes são mais antigos, porém isto garante a estabilidade do sistema, item este que é o grande foco para aplicação em servidores. Por conta da estabilidade e facilidade de uso do Debian, muitas distribuições comerciais se baseiam no Debian, incluindo: Linspire (antigo Lindows), Xandros, Knoppix, Kurumin, BrDesktop e Ubuntu. Esforços têm sido feitos para portar o Debian para outros núcleos livres além do Linux, incluindo o Hurd e o BSD. Por enquanto, ainda é mais preciso citar o Debian como uma "Distribuição Linux", sem outras qualificações (**??**).

4.4.6 APACHE HTTP SERVER

Criado em 1995 por Rob McCool, então funcionário do NCSA (National Center for Supercomputing Applications), o servidor Apache (Servidor HTTP Apache - *Apache HTTP Server*) é reconhecido pela comunidade da computação como o mais bem sucedido servidor Web código livre. Em pesquisa realizada em dezembro de 2007, foi constatado que a utilização do Apache é de $47,20\%$ dos servidores no mundo (WEB..., 2007). Em maio de 2010, o Apache serviu mais de $54,68\%$ de todos os sites e mais de 66% dos milhões de sites mais movimentados. É a principal tecnologia da *Apache Software Foundation*, responsável por mais de uma dezena de projetos envolvendo tecnologias de transmissão via Web, processamento de dados e execução de aplicativos distribuídos (WEB..., 2010).

Além disso, o Apache é compatível com o protocolo HTTP versão 1.1 e funcionalidades são mantidas através de uma estrutura de módulos, permitindo inclusive que o usuário escreva seus próprios módulos — utilizando a API do software. É disponibilizado em versões para a grande maioria dos sistemas operacionais como: Windows, Novell Netware, OS/2 e diversos outros do padrão POSIX (Unix, Linux, FreeBSD, etc.) (WHY..., 2012a).

Quanto ao Apache ser gratuiro, de acordo com a Apache Software Foudation, o aplicativo Apache existe para fornecer implementações de referência robustos e de nível comercial de muitos tipos de software. Deve continuar a ser uma plataforma na qual os indivíduos, assim as instituições podem construir sistemas confiáveis, tanto para fins experimentais quanto de missão crítica. O grupo arcedita que as ferramentas de publicação *on-line* devem estar nas mãos de todas as pessoas, e que as empresas de software devem utilizar o seu dinheiro para fornecer serviços de valor agregado, tais como módulos especializados e de apoio, entre outras. As empresas em geral veem tudo isto como uma vantagem econômica de modo que a empresa que possuir"um mercado - na indústria de software - significa controlar firmemente o canal de tal forma que todos devem pagar por seu uso. Isso geralmente é feito por "possuir"os meios pelos quais as empresas realizam negócios, beneficiando-se à custa de todas essas outras empresas. Na medida em que os protocolos da *World Wide Web* permanecer "sem dono"por uma única empresa, a Web continuará a possuir concorrências equitativas para as empresas grandes e pequenas. Assim, a Apache Foundation acredita que a "propriedade"dos protocolos deve ser evitada. Para este fim, a existência de implementações de referência robustas de vários protocolos e interfaces de programação de aplicativo devem estar disponíveis gratuitamente para todas as empresas e indivíduos (WHY..., 2012a).

Além disso, a Apache Software Foundation que constrói uma cultura em que aqueles que se beneficiam deste software usando-o, muitas vezes contribuem de volta a ele, fornecendo

aprimoramentos de recursos, correções de bugs, e suporte para os outros nas listas públicas e *newsgroups*. O esforço exercido por qualquer indivíduo em particular é relativamente pequeno, mas o produto resultante disto possui um resultado muito forte. Esses tipos de comunidades só pode acontecer com software livre — quando alguém paga por um software, as empresas geralmente não estão dispostas a corrigir os seus erros de forma gratuita. Assim, pode-se dizer então que a força do Apache vem do fato de que ele é livre e, se fosse feito "não livre", sofreria bastante para crescer, mesmo que os recursos fosse gasto em uma equipe de desenvolvimento real (WHY. . . , 2012a). Considerando as características descritas e visto que o Apache é construído para aplicações robustas baseadas em software livre, tudo isto torna o Apache HTTP Server uma ferramenta importante e justa para uso na implementação do projeto.

4.5 ARQUITETURA DA SOLUÇÃO

A arquitetura para a solução apresentada é apresentada na Figura 4. Toda a arquitetura é composta de algumas partes importantes, o *array de discos*, *as máquinas targets*, *o switch intermediário entre a máquina initiator e target*, tudo isto formando uma única rede SAN; e, recebendo os recursos, uma *máquina initiator* contendo um servidor web que funcionará como um *gateway* (aplicativo de interfaceamento) para requisições de dados e compartilhamento à rede SAN.

Cada *array de discos* é formado por um aglomerado de discos baratos que são conectados logicamente através de RAID5, o que aumenta a performance e garante a integridade dos dados caso se perca um dos discos. Sobre este conjunto de discos, a *máquina target* utiliza o LVM (*Logical Volume Manager*), que facilita o gerenciamento dos discos que são apresentados à *máquina initiator* através do protocolo AoE. Incremento e decremento de espaço e acréscimo de outros arrays de disco no mesmo volume lógico são exemplos dos benefícios que se podem extrair do LVM

Todas as máquinas são ligadas ao *switch layer 2* através de cabos par trançados com conectores RJ-45 em suas pontas formando uma única rede SAN (tráfego exclusivo de dados). Neste ponto todos os *arrays de discos* exportados pelas *máquinas target* são enxergados pela *máquina initiator* como apenas um único disco, ficando para as *máquinas target* a tarefa de gerenciar os discos individuais. A *máquina initiator*, ao receber cada dispositivo de bloco dos *targets*, monta todos dentro de mais um nível de volume lógico (LVM), dando mais flexibilidade de manutenção para toda a arquitetura organizacional dos discos. Isto é importante para facilitar o gerenciamento de toda a solução de *storage*.

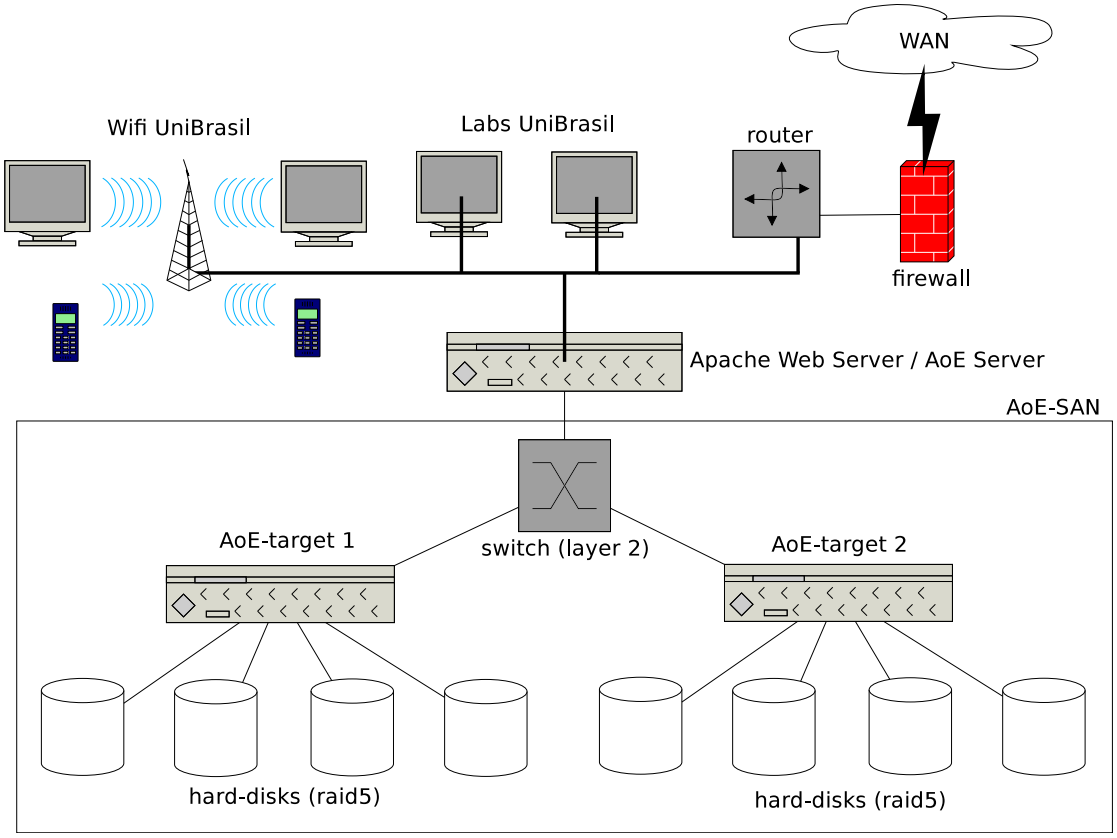


Figura 4: Arquitetura da Solução.

O *gateway* feito pelo servidor Web faz o interfaceamento entre a camada de armazenamento e a apresentação para o usuário. Além disso, é responsável por toda a lógica de acesso aos dados e políticas de permissões de acesso aos usuários finais. A aplicação Web que provê o acesso aos usuários funciona de modo semelhante a serviços como *4Shared* (4SHARED..., 2005) e *Sourceforge* (SOURCEFORGE..., 2011).

Por fim, as camadas acima do servidor Web mostradas na Figura 4 representam a atual rede da UniBrasil, o que significa que todos os usuários que utilizam a rede e possuam permissões de acesso aos arquivos no *gateway* poderiam fazer uso da ferramenta para compartilhar seus dados. Caso houvesse um redirecionamento no *firewall* (ou *gateway*) da faculdade, os mesmos usuários poderiam ter acesso a esta infraestrutura através da Internet.

4.6 ANÁLISE DE DESEMPENHO ENTRE OS PROTOCOLOS

Todas as análises realizadas são baseadas no artigo submetido ao SEMISH (Seminário Integrado de Software e Hardware), evendo da SBC (Sociedade Brasileira de Computação) feito pelo Msc. Pedro Eugênio Rocha e pelo autor deste TCC Leonardo Antônio dos Santos; o qual fundamentou o porquê de se utilizar o protocolo AoE em relação ao iSCSI neste trabalho.

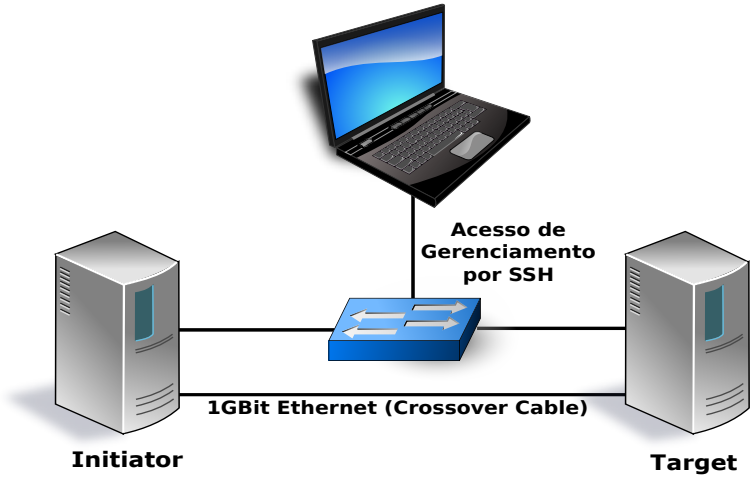


Figura 5: Cenário de testes.

4.6.1 OUTROS TRABALHOS DE TESTES

A popularização da virtualização e sua aplicação na consolidação de ambientes para *Cloud Computing* aumentou a demanda por protocolos eficientes de armazenamento de dados em rede. Apesar de existirem diferentes alternativas, como FC, iSCSI e AoE, o protocolo a ser utilizado depende fortemente de requisitos como desempenho, confiabilidade, latência e, principalmente, custo.

Grande parte das avaliações de desempenho de protocolos de armazenamento em SANs presentes na literatura restringem-se à análise individual de um protocolo (TAN et al., 2005; AIKEN et al., 2003; ZHOU; CHUANG, 2009). Embora existam comparações de desempenho entre diferentes protocolos, muitos dos trabalhos comparam *Fibre Channel*, por ser amplamente utilizado em sistemas organizacionais de grande porte, com protocolos alternativos (VORUGANTI; SARKAR, 2001; FOLLETT, 2001). Contudo, o protocolo *Fibre Channel* exige a utilização de especializado e de alto custo, fugindo do escopo deste trabalho.

Uma análise preliminar do desempenho dos protocolos AoE e iSCSI é mostrada em (CHUANG; WENBI, 2009). Neste artigo, microbenchmarks de escrita são executados sobre ambos os protocolos, variando o MTU dos quadros Ethernet. Além disso, a utilização de processamento dos protocolos é comparada. Em (GERDELAN et al., 2007), é apresentada mais uma comparação entre os protocolos iSCSI e AoE, considerados *eficientes em termos de custo* por não empregarem hardware especializado. Gerdelan *et al.* também analisam o desempenho dos protocolos somente através da comparação dos resultados de diferentes microbenchmarks em ambas as arquiteturas.

Argumentamos que, apesar de fornecer uma ideia preliminar do desempenho, micro-

benchmarks não refletem o comportamento dos protocolos em cenários reais de uso. Uma medição mais precisa deve considerar os diferentes tipos de workloads que podem existir em sistemas reais e de grande porte, que podem ser simulados com a utilização de macrobenchmarks.

4.6.2 AMBIENTE DE TESTES E METODOLOGIA

Com o objetivo de testar o desempenho dos protocolos de armazenamento de rede, montamos um ambiente contendo dois servidores. O primeiro servidor, chamado de *target*, contém dois processadores Xeon X5690 *six-core* 3.47 GHz, 64 GB de memória RAM e discos 10.000 rpm SCSI com capacidade de 300 GB. Um disco é utilizado exclusivamente para os testes com os protocolos. A segunda máquina, ou *initiator*, contém dois processadores Xeon *dual-core* 3 GHz, 8 GB de memória RAM e discos SCSI de 146 GB. As duas máquinas possuem mais de uma interface de rede, sendo diretamente interligadas através de interfaces Ethernet 1 Gigabit dedicadas para os testes. Finalmente, ambas utilizam o sistema operacional Debian 6.0.

Primeiramente, executamos um conjunto de microbenchmarks sobre o ambiente de testes criado. Os microbenchmarks têm o objetivo de estabelecer uma linha de base sobre o desempenho esperado dos protocolos em situações muito específicas de uso. Os resultados obtidos através dos microbenchmarks, embora não reflitam situações próximas das reais de uso, são úteis na interpretação de resultados mais complexos, como os obtidos na execução de workloads reais.

Em seguida, executamos um conjunto de macrobenchmarks sobre o mesmo ambiente de testes. Os macrobenchmarks têm o objetivo de medir o desempenho de diferentes configurações do sistema em workloads muito próximos dos encontrados em ambientes reais. Tais workloads, embora sintéticos, simulam os workloads encontrados em sistemas como servidores Web, servidores de arquivos, servidores de e-mail e bancos de dados transacionais, por exemplo. Além disso, alguns parâmetros como quantidade de memória disponível e MTU da interface de rede são alterados, visando verificar sua influência na vazão de disco alcançada por cada protocolo. Com base nestes resultados, analisaremos a eficiência dos protocolos em cada situação, bem como o impacto que a escolha do protocolo pode apresentar sobre o desempenho geral do sistema.

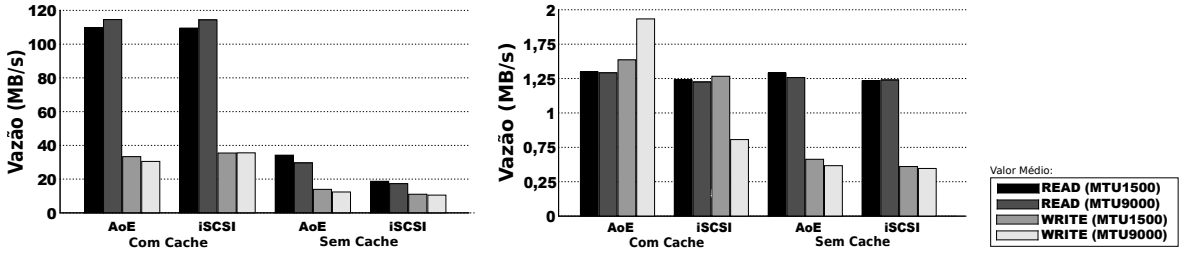
Após apresentar os resultados obtidos pelos macrobenchmarks, uma análise em termos de vazão, utilização de CPU e de rede é mostrada, relacionando-os com os obtidos pelos microbenchmarks. Por fim, uma discussão sobre os resultados encontrados é apresentada, enumerando algumas importantes observações sobre o desempenho dos protocolos, que, até onde

sabemos, inexiste em trabalhos anteriores.

4.6.3 MICROBENCHMARKS

Nesta Seção, apresentamos os resultados obtidos através da execução de microbenchmarks utilizando os protocolos AoE e iSCSI. Os resultados estão divididos entre operações de leitura e escrita, padrão de acesso sequencial e aleatório, utilização ou não de caches e MTU da interface de rede (1500 e 9000 bytes). Todos os microbenchmarks foram executados através da ferramenta *fio*, que possibilita a emissão de diversos padrões de acesso de I/O e diferentes parâmetros de forma customizada e flexível.

Os resultados dos experimentos são ilustrados na Figura 6. A Figura (a) mostra os resultados dos testes de leitura e escrita sequencial enquanto a Figura (b) mostra os resultados obtidos em leitura e escrita aleatória, sob diferentes configurações. Os resultados apresentados no microbenchmark correspondem à vazão média alcançada em três execuções.



(a) Acesso Sequencial (b) Acesso Aleatório

Figura 6: Vazão dos disco alcançada com diferentes parâmetros de microbenchmark.

De acordo com os gráficos, é possível observar que o protocolo AoE apresenta melhor desempenho no teste de leitura sequencial sem cache, cerca de 20%, independentemente do MTU. A perda de desempenho do protocolo iSCSI é atribuído ao *overhead* causado pelo processamento das camadas de rede adicionais, quando comparado ao AoE, que opera diretamente na camada de enlace. Todavia, quando o padrão de acesso é leitura aleatória, ainda sem cache, ambos apresentam o mesmo resultado. Neste caso, o *overhead* causado pelo acesso aleatório ao disco (*seek time*) na máquina *target* é predominante, suavizando o *overhead* causado pelas camadas de rede.

Por outro lado, quando os testes de leitura são executados com cache, ambos os protocolos apresentam grande melhora de desempenho quando executados de forma sequencial, devido ao mecanismo de *read-ahead* do sistema operacional. Pelo mesmo motivo, como testes com padrão aleatório não se beneficiam com *read-ahead*, nenhuma alteração no desempenho é obtida pela adição de caches nestes padrões de acesso.

O mesmo comportamento observado na leitura sequencial e sem caches ocorre nas escritas — o protocolo AoE apresenta melhor desempenho devido ao *overhead* das camadas de rede. Contudo, diferentemente das leituras, o protocolo AoE apresenta melhor desempenho mesmo em padrões de acesso aleatórios. Como escritas são, em geral, assíncronas, de forma que os dados são escritos apenas em memória na máquina *target* e posteriormente escritos em disco, o tempo de acesso aleatório ao disco é mitigado, tornando novamente significativo o *overhead* das camadas de rede do iSCSI, diminuindo assim seu desempenho em relação ao protocolo AoE.

Quando as escritas são executadas com cache, por outro lado, o protocolo iSCSI apresenta desempenho superior em todos os experimentos. Isso ocorre pois o subsistema SCSI da máquina *initiator* implementa a política de cache conhecida como *write-back*, onde os dados são escritos no cache da máquina local e enviados à maquina remota em momento oportuno. Assim, o alto *desempenho percebido* pelo protocolo iSCSI em testes de escrita com cache é maior que o *desempenho real*, na medida em que as operações de escritas não atingem imediatamente o disco da máquina destino.

4.6.4 MACROBENCHMARKS

Para analisar o comportamento de ambos os protocolos de armazenamento de redes SAN em situações próximas das encontradas em ambientes reais, uma sequência de macrobenchmarks foi executada. Para tal, utilizamos a ferramenta de benchmarks *filebench* por ser amplamente utilizada e por conter diversos workloads pré-definidos, simulando o padrão real de acesso em servidores com diferentes finalidades. Os workloads pré-definidos utilizados neste experimento são descritos abaixo.

Webserver: Simula o padrão de acesso de um servidor Web. O workload, predominantemente de leitura, executa um conjunto de operações do tipo *open-read-close* em diferentes arquivos de 16 KB, contando com um total de 50.000 arquivos. Além disso, há um fluxo de operações do tipo *append* que simula a escrita em arquivos de log.

Fileserver: Emula o funcionamento de um servidor de arquivos. O workload consiste em uma sequência de operações de criação, exclusão, escrita, leitura e operações sobre metadados em um conjunto de 10.000 arquivos, com tamanho de, em média, 128 KB.

Varmail: Simula a atividade de I/O de um servidor de e-mails que armazena cada mensagem como um arquivo individual. O workload consiste em uma sequência de operações do tipo *create-append-sync*, *read-append-sync*, leitura e exclusão de arquivos em um mesmo

diretório.

Oltp: Emula o padrão de acesso de um banco de dados transacional. Este workload testa o desempenho de múltiplas operações aleatórias de leitura e escrita sensíveis à latência. O padrão de acesso é baseado no banco de dados Oracle 9i.

Em todos os workloads, o número de threads foi ajustado de forma a não sobrecarregar a CPU e influenciar o resultado obtido pelos testes. Nas Subseções seguintes, apresentamos os resultados obtidos através dos experimentos sob três dimensões: vazão no acesso ao disco remoto (Subseção 4.6.5), utilização de CPU (Subseção 4.6.6) e utilização de rede (Subseção 4.6.7).

4.6.5 VAZÃO DE DISCO

A Figura 7 apresenta os resultados obtidos através dos diferentes workloads. Nos gráficos, o eixo horizontal mostra a execução dos experimentos no disco local e utilizando os protocolos de armazenamento remoto. Para cada grupo de barras, são apresentadas as execuções com quantidades diferentes de memória disponível ao sistema (512 MB e 4 GB) e MTU (1500 e 9000 bytes). Para os testes de acesso local, apenas a quantidade de memória é alterada. Limitamos a quantidade de memória disponível ao sistema para amenizar o efeito das caches no resultado obtido, visto que são implementadas em diversos níveis do sistema operacional, como caches de blocos, páginas e mesmo dentro do subsistema SCSI. Mesmo assim, durante todos os experimentos houve memória suficiente às aplicações, não havendo uso de *swap*. O eixo vertical apresenta a vazão obtida.

Em workloads de leitura predominante, como o *webserver*, mostrado na Figura (a), o uso de caches atenua a diferença de vazão entre os protocolos. Como esperado, este resultado segue o comportamento encontrado na execução dos microbenchmarks de leitura sequencial. Em todos os casos de teste, invariavelmente, ao fornecer memória RAM suficiente ao sistema (4 GB), a variação dos resultados entre os protocolos é desprezível — inferior a 1%. Caso a quantidade de memória disponível ao sistema seja escassa, simulado pelo caso de teste de 512 MB, é possível observar que o protocolo AoE possui melhor desempenho, em torno de 8%, devido ao *overhead* de processamento das camadas rede. Entretanto, considerando a quantidade e tamanho dos arquivos criados pelo benchmark, e que servidores atuais possuem, em geral, mais do que 4 GB de memória RAM, afirmamos empiricamente que *o uso de cache iguala o desempenho dos protocolos em workloads de leitura predominante*.

Já em workloads que executam massivas operações de escrita, como é o caso do *file-*

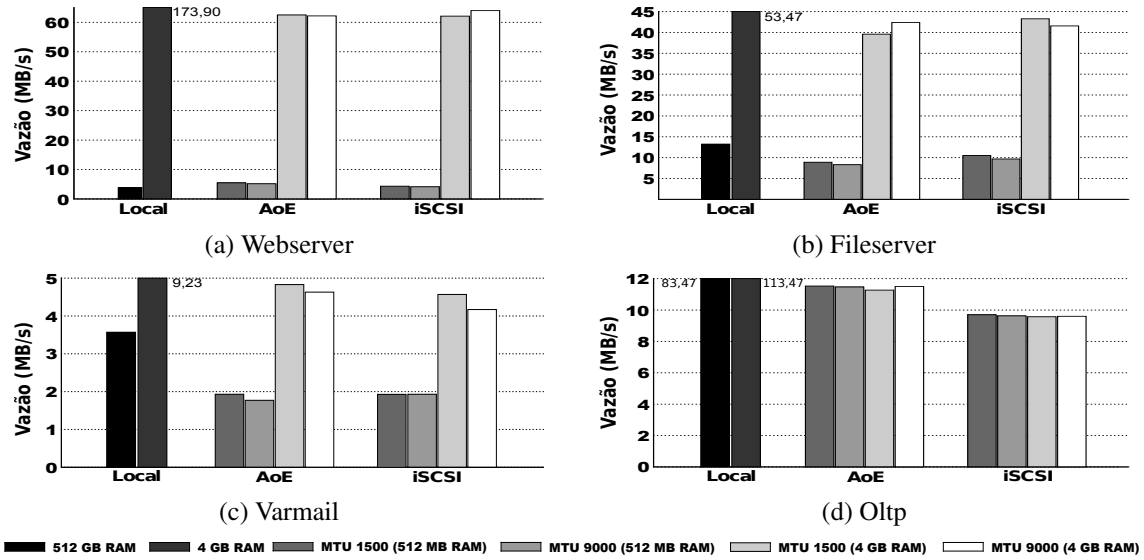


Figura 7: Vazão de disco alcançada por diferentes tipos de workloads em cenários de execução diversos. Onde indicado, o valor correto das barras foi modificado para melhor visualização dos resultados.

server, o protocolo iSCSI apresenta melhor desempenho, em média 9%, independentemente do MTU empregado. A utilização da política de cache *write-back* nas escritas, onde as operações são efetuadas localmente na memória da máquina *initiator* e depois executadas remotamente em momento oportuno, garante que o desempenho percebido seja superior ao do protocolo AoE. Nos casos testados, quanto maior a quantidade de memória, e consequentemente o espaço disponível para cache, maior é o aumento na vazão. Assim, concluímos que *o protocolo iSCSI apresenta melhor desempenho em workloads de escrita predominante, caso haja espaço suficiente para cache.*

Quando o workload consiste em um número balanceado de operações de escrita e leitura, como no workload *varmail*, o protocolo AoE mostra desempenho $11,2\%$ superior. Por um lado, a quantidade de operações de escrita não é suficientemente grande para que o iSCSI beneficie-se de seu melhor desempenho em escritas; por outro, as operações de leitura não são exclusivamente sequenciais (já que as operações ocorrem sobre múltiplos arquivos). Assim, o *overhead* do protocolo iSCSI torna-se determinante na vazão alcançada por este experimento.

Por fim, no caso do workload *oltp*, onde o padrão predominante é a escrita aleatória, o protocolo AoE apresenta melhor desempenho em todos os resultados, em média 19%. Novamente, este resultado é semelhante ao encontrado nos testes com microbenchmarks. Todavia, é importante ressaltar que, apesar do aumento na quantidade de memória disponível para caches, não houve aumento significativo na vazão. Isso ocorre porque parte das operações executados pelo workload são realizadas sem cache (utilizando a flag O_DIRECT). Assim, notamos que em todas as configurações deste workload o protocolo AoE obteve melhor desempenho.

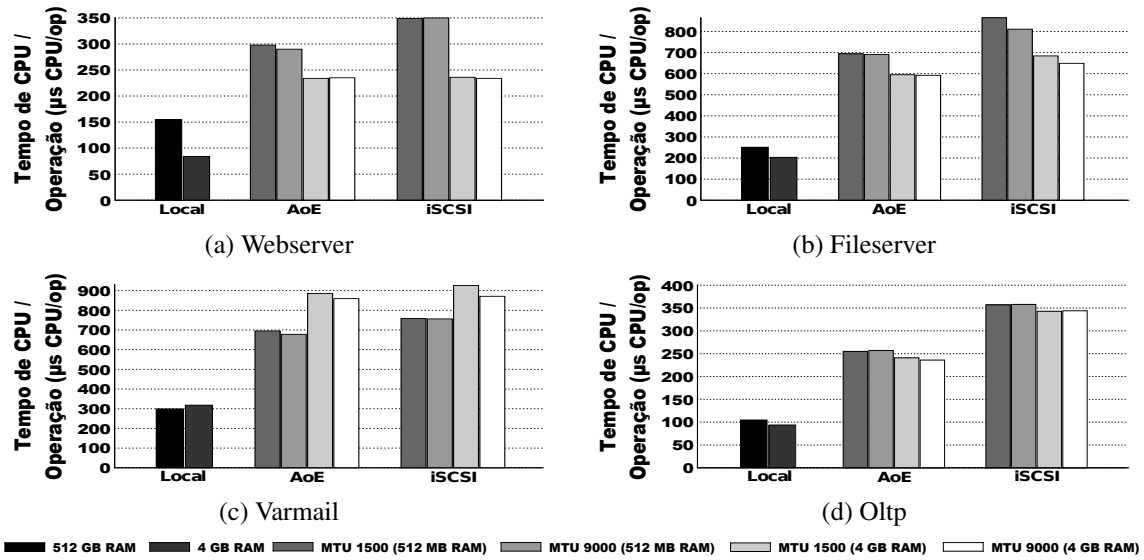


Figura 8: Tempo de CPU por operação de I/O em diferentes workloads.

4.6.6 UTILIZAÇÃO DE CPU

A Figura 8 apresenta a utilização de CPU em cada caso de teste. O eixo horizontal apresenta os mesmos grupos de barras do teste anterior; o eixo vertical, a média do tempo de CPU gasto por cada operação de I/O emitida pelo benchmark. Quanto menor o tempo de processamento, mais eficiente em termos de CPU o protocolo é considerado.

Através dos resultados, é possível observar que o protocolo iSCSI apresenta maior utilização de CPU em todos os testes realizados. Este comportamento é esperado devido à camada em o que o protocolo opera, pois acrescenta o *overhead* de processamento das camadas de transporte e rede, quando comparado ao protocolo AoE, que opera diretamente na camada de enlace. Este fato pode ser percebido em todos os casos, e independe do resultado da vazão alcançada pelos protocolos. Entretanto, apesar do protocolo iSCSI apresentar maior utilização de CPU em todos os casos, o uso de caches diminui significativamente a diferença entre a utilização de CPU dos protocolos, de $22,2\%$ sem cache para 15% com cache, na medida em que distribui o *overhead* das operações custosas realizadas no servidor remoto entre as operações que acertaram a cache local. Assim, o uso de caches diminui significativamente a quantidade de processamento necessária por operação, aumentando a eficiência de CPU de ambos os protocolos.

Além disso, pode-se observar que em grande parte dos testes, como esperado, aumentar o MTU da interface de rede pode diminuir o tempo de CPU gasto por operação pelos dois protocolos. Isso ocorre pois, como o número de quadros Ethernet enviados é menor, menor é o tempo de processamento necessário para interpretá-lo.

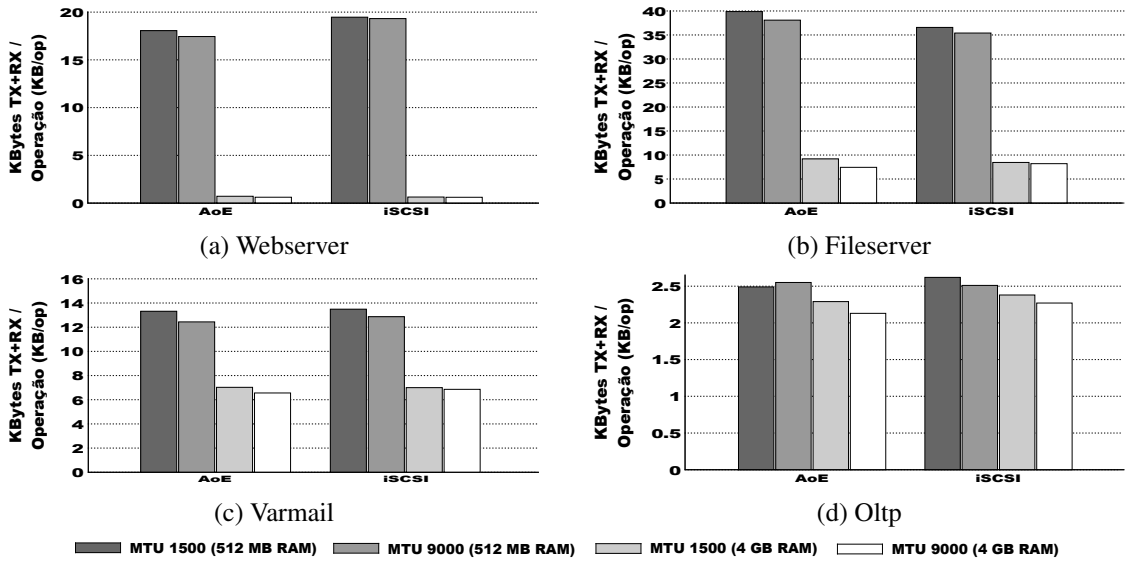


Figura 9: Quantidade de dados trafegado na rede (transmitido + recebido) por operação.

4.6.7 UTILIZAÇÃO DE REDE

A Figura 9 apresenta a quantidade total de bytes transmitidos e recebidos divididos pelo número total de operações realizadas em cada caso de teste. O eixo horizontal apresenta os mesmos grupos de barras dos testes anteriores; o eixo vertical, o número médio de bytes por operação realizada pelo benchmark. É importante notar que, como parte das operações é executada em cache sem utilizar a interface de rede, o valor apresentado não reflete a quantidade real de rede utilizada por operação; entretanto, o valor é utilizado como métrica para comparação, na medida em que reflete a eficiência em termos de rede dos protocolos sob diferentes configurações.

Como esperado, em todos os casos, quanto maior a quantidade de memória disponível ao sistema, menor é a quantidade de dados trafegados na rede, devido à maior quantidade de operações realizadas em cache local. Da mesma forma, na maioria dos casos testados o AoE possui melhor eficiência de uso de rede (média de 3% para Webserver, Varmail e Oltp), devido a ausência do *overhead* introduzido pelas camadas adicionais no iSCSI, com uma exceção: no workload *fileserver*, como o iSCSI utiliza o cache com mais eficiência na escrita, consequentemente sua utilização de rede é menor ($3,26\%$).

Além disso, na maioria dos casos, aumentar o MTU na interface de rede melhora a eficiência na utilização de rede, pois mais dados são enviados em um mesmo quadro Ethernet, diminuindo assim o *overhead* introduzido pelos cabeçalhos. Nos poucos casos em que ocorre o contrário, memória RAM limitada é utilizada (512 MB); nesses, acreditamos que a alocação de buffers maiores para os quadros Ethernet diminua a quantidade de memória disponível para

a cache de disco, diminuindo a taxa de acerto em cache e portanto aumentando a utilização de rede.

4.6.8 DISCUSSÃO

Através do conjunto de experimentos executados, observamos que o protocolo AoE apresenta melhor desempenho global, de cerca de 9%. Contudo, ao fornecer quantidade suficiente de memória ao sistema, o protocolo iSCSI alcança melhor resultado em workloads de escrita predominante, devido a mecanismos inteligentes de cache, e resultado muito próximo ao AoE, mas ainda inferior, em workloads de leitura. Ainda assim, é possível afirmar que o protocolo iSCSI é uma boa alternativa para a maioria dos workloads, considerando que servidores atuais possuem quantidade considerável de memória RAM e que o protocolo iSCSI possui outras vantagens, como o fato de ser roteável e possibilidade de utilização de criptografia, através de IPSec.

Apesar disso, em workloads que não utilizam exaustivamente o mecanismo de cache do sistema operacional, como o Oltp, em que as caches são geralmente gerenciadas pela própria aplicação, o iSCSI também apresenta desempenho inferior. Nestes casos, sob todas as dimensões (vazão, CPU e rede) e protocolo AoE apresenta melhor desempenho, sendo, definitivamente, a melhor solução.

Outro fato a ser considerado é que o protocolo iSCSI apresenta, invariavelmente, *overhead* de CPU de cerca de 19% quando comparado ao AoE. Assim, caso haja restrições de processamento, a utilização do protocolo iSCSI deve ser evitada. Quanto à utilização de rede, a eficiência do protocolo iSCSI é cerca de 3% menor se comparado ao AoE; entretanto, considerando os fatos observados e a capacidade das redes atuais (1, 10 e até 40 Gbps), em poucos casos a capacidade da rede pode limitar a vazão no acesso ao disco. Finalmente, o aumento do MTU dos quadros Ethernet melhora a vazão em todos os casos, além de diminuir a quantidade de processamento e a utilização de rede. O aumento do MTU deve ser evitado somente quando a quantidade de memória for limitada, onde os buffers alocados para os quadros maiores possam concorrer com o mecanismo de cache de disco.

Estes protocolos, além de amplamente utilizados em SANs de ambientes de computação em nuvem e virtualização, não necessitam de hardware especializado e possuem implementações de código livre, diferentemente do protocolo FC. Através de um conjunto extensivo de experimentos, baseados tanto em microbenchmarks como em macrobenchmarks, analisamos as principais características desses protocolos em termos de vazão de dados, quantidade de memória cache, utilização de CPU e rede.

Observamos a partir de nossos experimentos que o protocolo iSCSI apresenta cerca de 9% de aumento na vazão em workloads de escrita predominante e desempenho muito próximo ao AoE em workloads de leitura (próximo a 1% no caso do webserver), caso haja memória suficiente para cache. Apesar disso, o AoE é a melhor opção para workloads que evitam o mecanismo de cache do sistema operacional, como bancos de dados Oltp. Ademais, o protocolo iSCSI é menos eficiente em termos de CPU e utilização de rede sob qualquer workload.

Dando continuidade a este trabalho, pretendemos analisar outros aspectos dos protocolos, como sua escalabilidade, confiabilidade na presença de falhas e interações com sistemas de arquivos. Acreditamos que nosso trabalho possa auxiliar administradores de infraestrutura a melhor entender o funcionamento destes protocolos sobre diferentes workloads, bem como suas implicações sobre utilização de memória, CPU e rede, ajudando-os a escolher o melhor protocolo para cada caso, além de dimensionar corretamente a quantidade de recursos necessários.

Serão executados testes de análise de performance sobre o protocolo, a rede, os discos e outras tecnologias envolvidas, como o servidor Web e o servidor de banco de dados.

5 CONCLUSÃO

Este trabalho visa facilitar o uso de um protocolo código-aberto para o compartilhamento de arquivos em alta disponibilidade, tornando esta tarefa acessível a qualquer pessoa que assim o desejar.

Com a crescente demanda sobre esta área, principalmente no que se refere à virtualização de servidores, o que necessita de um armazenamento distribuído, este trabalho tenta trazer simplicidade ao uso de protocolos de armazenamento de dados para estas novas tendências, em especial por este recurso poder ser utilizado por qualquer pessoa, por conta de sua característica de software livre.

Para trabalhos futuros, podem ser efetuadas mudanças no próprio protocolo, visando o aumento da sua performance em situações específicas, como no caso da sua utilização em conjunto com RAID (*Redundant Array of Independent Drives*).

6 CRONOGRAMA

O cronograma 10 representa apenas uma hipótese temporal projeto a ser desenvolvido, podendo ser adaptado ao longo do projeto.

Bimestre / Atividades	2011		2012				
	Ago./ Set.	Out./ Nov.	Fev./ Mar.	Abr./ Mai.	Jun./ Jul.	Ago./ Set.	Out./ Nov.
Formulação do Projeto de Pesquisa	X						
Apresentação do Pré-projeto		X					
Levantamento Bibliográfico	X	X	X	X	X	X	
Levantamento de Requisitos		X	X				
Análise e compilação dos dados e informações.			X	X			
Planejamento da solução em informática.		X	X	X	X		
Apresentação do TCC1 em banca de qualificação					X		
Desenvolvimento da solução em informática				X	X	X	
Teste piloto da solução desenvolvida.				X	X	X	
Apresentação pública do TCC2 em banca de defesa.							X
Confecção de um artigo científico – Grupo de pesquisa – SI.					X	X	

Figura 10: Cronograma do Projeto.

REFERÊNCIAS

4SHARED.COM - free file sharing and storage. [S.l.]: 4shared.com, 2005. http://www.4shared.com/. Acesso em: 20 set. 2011.

AIKEN, S. et al. A performance analysis of the iscsi protocol. In: **Mass Storage Systems and Technologies, 2003. (MSST 2003). Proceedings. 20th IEEE/11th NASA Goddard Conference on**. [S.l.: s.n.], 2003. p. 123 – 134.

AMBLER, S. W. **Mapping Objects to Relational Databases: O/R Mapping In Detail**. [S.l.]: Ambysoft Inc, 2002. http://www.agiledata.org/essays/mappingObjects.html. Acesso em: 19 set. 2011.

ATA over Ethernet Tools. [S.l.]: Coraid, 2012. http://aoetools.sourceforge.net/. Acesso em: 15 mai. 2012.

CAKEPHP: the rapid development php framework. [S.l.]: Cake Software Foundation Â©, Inc, 2005. http://cakephp.org/. Acesso em: 19 set. 2011.

CASE Studies. [S.l.]: Oracle Corporation and/or its affiliates, 2012. http://www.mysql.com/why-mysql/case-studies/. Acesso em: 14 set. 2011.

CHUANG, H.; WENBI, R. **Modeling and Performance Evaluation of the AoE Protocol**. [S.l.]: IEEE Press, 2009. 609–6012 p.

CLARK, T. **Designing storage area networks: a practical reference for implementing Fibre Channel SANs**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999. ISBN 0-201-61584-3.

FASHEH, M. **OCFS2: The Oracle Clustered File System, Version 2**. 2006. Maio de 2007: http://oss.oracle.com/projects/ocfs2/dist/documentation/fasheh.pdf.

FOLLETT, D. **Distributed storage networking architectures: fibre channel, iSCSI, ethernet, infiniband, hyper transport, rapid IO and 3GIO collide in the data center**. 2001. 159 p.

FUNDAMENTALS of Networked Storage. [S.l.]: Coraid, Inc, 2008. http://www.vmworld.com/docs/DOC-1374.

GERDELAN, A.; JOHNSON, M.; MESSOM, C. Performance analysis of virtualised head nodes utilising cost-effective network attached storage. In: **Proceedings of the Asian Particle Accelerator Conference**. [S.l.: s.n.], 2007.

HOPKINS, S.; COILE, B. **ATA over Ethernet Specification**. [S.l.]: The Brantley Coile Company, Inc, 2001. http://support.coraid.com/documents/AoEr11.txt.

LANDOWSKI, M.; CURRAN, P. F. Aoe storage protocol over mpls network. In: **Proceedings of the 2011 IEEE 27th Symposium on Mass Storage Systems and Technologies**. Washington, DC, USA: IEEE Computer Society, 2011. (MSST '11), p. 1–5. ISBN 978-1-4577-0427-7. Disponível em: <http://dx.doi.org/10.1109/MSST.2011.5937231>.

LEWIS, A. **LVM HOWTO**. [S.l.]: The Linux Documentation Project, 2006. `http://www.tldp.org/HOWTO/LVM-HOWTO/`.

LOBUE, M. T. et al. Surveying today's most popular storage interfaces. **Computer**, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 35, n. 12, p. 48–55, dez. 2002. ISSN 0018-9162. Disponível em: <http://dx.doi.org/10.1109/MC.2002.1106179>.

PATTERSON, D. A.; GIBSON, G. A.; KATZ, R. H. A case for redundant arrays of inexpensive disks (raid). In: **SIGMOD Conference**. [S.l.: s.n.], 1988. p. 109–116.

PHP: Hypertext Preprocessor. [S.l.]: The PHP Group, 2001. `http://www.php.net/manual/en/index.php`. Acesso em: 14 set. 2011.

SATRAN, J. et al. **Internet Small Computer Systems Interface (iSCSI)**. IETF, abr. 2004. RFC 3720 (Proposed Standard). (Request for Comments, 3720). Updated by RFCs 3980, 4850, 5048. Disponível em: <http://www.ietf.org/rfc/rfc3720.txt>.

SOURCEFORGE.NET: Find, Create, and Publish Open Source software for free. [S.l.]: 2011 Geeknet, Inc, 2011. `http://sourceforge.net/`. Acesso em: 20 set. 2011.

TAN, Y. et al. A high-throughput fibre channel data communication service. In: **Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies.** [S.l.: s.n.], 2005. p. 975–978.

TANENBAUM, A. S. **Sistemas Operacionais Modernos**. trad. 3 ed. São Paulo: Pearson, 2010.

TRYGVE/MVC. 2011. `http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html`. Acesso em: 19 set. 2011.

VAGHANI, S. B. Virtual machine file system. **SIGOPS Oper. Syst. Rev.**, ACM, New York, NY, USA, v. 44, n. 4, p. 57–70, dez. 2010. ISSN 0163-5980. Disponível em: <http://doi.acm.org/10.1145/1899928.1899935>.

VORUGANTI, K.; SARKAR, P. An analysis of three gigabit networking protocols for storage area networks. In: **IEEE International Conference on Performance, Computing, and Communications, 2001.** [S.l.: s.n.], 2001. p. 259–265.

WEB Server Survey, 2007. [S.l.]: NetCraft Ltd, 2007. `http://news.netcraft.com/archives/2007/12/29/december_2007_web_server_survey.html`. Acesso em: 15 mai. 2012.

WEB Server Survey, 2010. [S.l.]: NetCraft Ltd, 2010. `http://news.netcraft.com/archives/2010/05/14/may_2010_web_server_survey.html`. Acesso em: 15 mai. 2012.

WHAT is the difference between SCSI and ATA? [S.l.]: insideHPC, LLC, 2006. `http://insidehpc.com/2006/04/07/what-is-the-difference-between-scsi-and-ata/`. Acesso em: 26 mar. 2012.

WHY Apache Software is Free. [S.l.]: Apache Software Foundation, 2012. http://httpd.apache.org/ABOUT_APACHE.html. Acesso em: 15 mai. 2012.

WHY MySQL? [S.l.]: Oracle Corporation and/or its affiliates, 2012. http://www.mysql.com/why-mysql/. Acesso em: 15 mai. 2012.

YOUTUBE - Broadcast Yourself. [S.l.]: 2011 YouTube, LLC, 2011. http://www.youtube.com/. Acesso em: 20 set. 2011.

ZHOU, C.; CHUANG, H. A performance analysis of the aoe protocol. In: **Proceedings of the 5th International Conference on Wireless communications, networking and mobile computing**. [S.l.]: IEEE Press, 2009. p. 3938–3941.