CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
GOVERNO DO ESTADO DE SÃO PAULO
EXTENSÃO DA ESCOLA TÉCNICA ESTADUAL
“Dr. DOMINGOS MINICUCCI FILHO” NA “EE CARDOSO DE ALMEIDA”
CURSO TÉCNICO EM INFORMÁTICA PARA INTERNET

LUIS HENRIQUE SOLER
WEBER G. H. NOGUEIRA

ENTENDENDO SERVIDORES WEB DE ALTA DEMANDA.
NODE.JS

BOTUCATU – SP
JULHO – 2011

CENTRO PAULA SOUZA
COMPETÊNCIA EM EDUCAÇÃO PÚBLICA PROFISSIONAL

CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

GOVERNO DO ESTADO DE SÃO PAULO

EXTENSÃO DA ESCOLA TÉCNICA ESTADUAL

“Dr. DOMINGOS MINICUCCI FILHO” NA “EE CARDOSO DE ALMEIDA”

CURSO TÉCNICO EM INFORMÁTICA PARA INTERNET

LUIS HENRIQUE SOLER

WEBER G. H. NOGUEIRA

ENTENDENDO SERVIDORES WEB DE ALTA DEMANDA. NODE.JS

Trabalho de Conclusão de Curso apresentado à Extensão da Escola Técnica Estadual “Dr. Domingos Minicucci Filho” na “E.E. Cardoso de Almeida”, sob colaboração do Profº Rodolfo Cristiano Serafim, para obtenção do título de Técnico em Informática para Internet.

BOTUCATU – SP

JULHO – 2011

FOLHA DE DEDICATÓRIA

“Aos professores e alunos.”

AGRADECIMENTOS

Ao Prof. Rodolfo Cristiano Serafim, nosso orientador. Aos professores que contribuíram ao longo do curso para nosso conhecimento, dedicando-se sempre em todos os momentos. Enfim, agradecemos a todos aqueles que de forma direta ou indireta contribuíram para a realização deste trabalho.

Agradecemos aos nossos amigos de turma do curso, pelo companheirismo e pelos ricos debates que realizamos ao decorrer do curso, bem como a todos aqueles que de forma direta ou indireta contribuíram para a realização desta monografia.

RESUMO

Na última década pudemos notar que o número de usuários da Internet saltou de aproximadamente 360 Milhões em 2000 para quase 2 Bilhões em 2010, representando um crescimento de 444,8%. Isso exige que os provedores de serviços na web invistam em estruturas que possam suportar um volume imprevisível de transações, ou seja, que possam ser expandidas sem muitas complicações. Nesse trabalho, comparamos duas soluções de servidores web identificando suas vantagens e desvantagens. Para isso foi analisada uma solução de servidor web de alta demanda baseada no Node.js, comparando-a com uma solução padrão de mercado baseada em Apache HTTP Server e PHP. Observamos que o Node.js obteve resultados melhores em relação ao desempenho, mas o Apache HTTP Server foi superior no quesito produtividade. Também identificamos a possibilidade de se integrar as duas soluções, utilizando-as em conjunto e delegando operações adequadas às características de cada tecnologia.

Palavras-chave: Node.js, PHP, Apache HTTP Server, Servidores Web de Alta Demanda, Tecnologia da Informação.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS OU SIGLAS

ab	ApacheBench
ECMA	European Computer Manufacturers Association
HTTP	Hypertext Transfer Protocol
I/O	Input/Output
ISO	International Organization for Standardization
JS	JavaScript
PHP	PHP: Hypertext Preprocessor

SUMÁRIO

1 INTRODUÇÃO

Na última década pudemos notar que o número de usuários da Internet continua crescendo: saltou de aproximadamente 360 Milhões em 2000 para quase 2 Bilhões em 2010, o que representa um crescimento de 444,8%. Isso nos indica que cerca de 29% da população mundial em 2010 já tem acesso à Internet (INTERNET WORLD STATS, 2011).

Esse crescimento exagerado deve-se a uma série de fatores, dos quais podemos destacar:

- Recursos de *Hardware* e *Software* que estão em constante processo de evolução e amadurecimento além da diminuição dos custos de implantação de tecnologias para Internet;
- Crescimento vertiginoso da diversidade de serviços baseados em *web* como redes sociais, portais de notícias, servidores de *e-mail*, jogos, etc.
- Aumento da quantidade de dispositivos com suporte a conexão com a Internet, como celulares, computadores portáteis, *tablets*, e até eletrodomésticos e veículos;
- Melhoria nas tecnologias de acessibilidade tanto para usuários quanto para outros sistemas;
- Diminuição dos custos de acesso à Internet banda larga, expansão da área de cobertura dos provedores de acesso e aumento da velocidade das conexões.

O aumento do número de usuários da Internet exige que os provedores de serviços na *web* invistam em estruturas que possam suportar um volume imprevisível de transações, ou seja, que possam ser expandidas sem muitas complicações.

Sistemas que precisam responder uma quantidade de acessos muito grande estão presentes em redes sociais, em jogos, em servidores de *download*, em serviços do governo, nos Correios, etc.

Em todos esses casos, há uma preocupação com a escalabilidade dos servidores, sendo que cada um aplica um modelo com tecnologias distintas, tanto de *hardware* quanto de *software*.

Em relação ao *hardware*, a escalabilidade pode ser feita de duas formas:

- **Vertical**: Consiste em melhorar o desempenho de um servidor único para que ele possa executar as tarefas mais rapidamente;
- **Horizontal**: Consiste em aumentar o número de máquinas (*Clusters*) trabalhando simultaneamente. Nesse caso, há a necessidade de que se configure também um tipo especial de servidor chamado de *proxy* reverso, responsável por intermediar a comunicação entre o cliente e o *Cluster* que processará a requisição.

Essas duas técnicas de escalabilidade (horizontal e vertical) podem ser aplicadas individual ou simultaneamente em um mesmo sistema.

A escalabilidade de *software* é feita através da otimização de alguns processos como:

- **Cache**: Armazenamento em disco de conteúdo estático para reduzir o número de operações de *I/O* desnecessárias.
- **Persistência de dados**: Armazenamento em memória de recursos que são utilizados frequentemente, para que o acesso se torne mais rápido do que o acesso a disco, rede ou banco de dados.
- **Finalizar requisições rapidamente**: Evitar filas de requisições aguardando por resposta.
- **Evitar *I/O* bloqueante**: Evitar que operações de entrada e saída bloqueiem o processamento de requisições concorrentes.
- **Segmentação**: transferir a responsabilidade de uma ou mais requisições para um segmento de processamento (*Thread*) individual.

A combinação do Apache HTTP Server (servidor *web*) com o PHP (Linguagem de programação) vem sendo usada amplamente para desenvolvimento *web*.

A documentação oficial do PHP resume seus detalhes e recursos:

> PHP, que significa "PHP: Hypertext Preprocessor", é uma linguagem de programação de ampla utilização, interpretada, que é especialmente interessante para desenvolvimento para a Web e pode ser mesclada dentro do código HTML. A sintaxe da linguagem lembra C, Java e Perl, e é fácil de aprender. O objetivo principal da linguagem é permitir a desenvolvedores escreverem páginas que serão geradas dinamicamente rapidamente, mas você pode fazer muito mais do que isso com PHP. (PHP DOCUMENTATION GROUP, 2011)

Segundo o *TIOBE Programming Community Index* em Maio de 2011, o PHP ocupa a quinta posição de popularidade de linguagens de programação, conforme a Tabela 1:

Tabela 1 – Popularidade das linguagens de programação

Position May 2011	Position May 2010	Programming Language	Ratings May 2011
1	2	Java	18,160%
2	1	C	16,170%
3	3	C++	9,146%
4	6	C#	7,539%
5	4	PHP	6,508%
6	10	Objective-C	5,010%
7	7	Python	4,583%
8	5	(Visual) Basic	4,496%
9	8	Perl	2,231%
10	11	Ruby	1,421%
11	12	JavaScript	1,394%
12	20	Lua	1,102%
13	9	Delphi	1,073%
14	-	Assembly	1,042%
15	16	Lisp	0,953%
16	23	Ada	0,747%
17	15	Pascal	0,709%
18	21	Transact-SQL	0,697%
19	-	Scheme	0,580%
20	25	RPG (OS/400)	0,503%

Fonte: TIOBE Software, 2011

A documentação oficial do Apache HTTP Server resume o propósito do *software*:

> O Apache HTTP Server é um servidor HTTP de código aberto que funciona em sistemas operacionais modernos como UNIX, Microsoft Windows, Mac OS/X e Netware. O objetivo do projeto é fornecer um servidor seguro, eficiente e extensível que disponibilize serviços HTTP com base nos padrões HTTP atuais. (THE APACHE SOFTWARE FOUNDATION, 2011, nossa tradução)

Segundo o site W3Techs, o Apache HTTP Server é usado em quase de 70% dos sites analisados por suas estatísticas, como descrito na Tabela 2:

Tabela 2 – Popularidade de softwares para servidores

Servidor	Participação
Apache	69,3%
Microsoft-IIS	19,4%
Nginx	7,0%
LiteSpeed	1,1%
Google Servers	0,8%
Tomcat	0,6%
Lighttpd	0,6%
IBM Servers	0,3%
Oracle Servers	0,2%
Yahoo Traffic Server	0,2%
Zeus	0,1%
Resin	0,1%
Zope	0,1%
Jetty	0,1%
Mongrel	0,1%

Fonte: W3Techs, 2011

As funcionalidades e objetivos do Node.js não estão muito claros no *site* oficial do *software*, mas um resumo pode ser encontrado em seu artigo na Wikpedia:

> Node.js é um framework orientado a eventos para o motor de JavaScript V8 que roda em plataformas Unix-like. Ele é usado para a criação de aplicações de rede escaláveis, como servidores web. Foi criado em 2009 por Ryan Dahl.
> O Node.js tem um propósito similar ao do Twisted (para Python), do Perl Object Environment (para Perl), e do EventMachine (para Ruby). Ao contrário da maioria dos códigos JavaScript, o código do Node.js não é executado em um navegador web, mas está atrelado à execução no lado do servidor. O Node.js implementa algumas das especificações do CommonJS. (WIKIMEDIA FOUNDATION, 2011)

2 OBJETIVO

2.1 Geral

Nesse estudo iremos analisar uma solução de servidor *web* de alto desempenho baseada no Node.js, comparando-a com uma solução padrão de mercado baseada em Apache HTTP Server e PHP.

2.2 Específico

Será feito um teste prático de desempenho das duas soluções, ambas rodando em condições semelhantes e com suas respectivas configurações padrão.

3 JUSTIFICATIVA

Nota-se que o número de usuários da Internet saltou de aproximadamente 360 Milhões em 2000 para quase 2 Bilhões em 2010. Esse aumento exige que os provedores de serviços na *web* invistam em novas estruturas e tecnologias que possam suportar um volume imprevisível de transações, ou seja, que possam ser expandidas sem muitas complicações. O tema foi escolhido, pois é notório que a questão da otimização de servidores é relevante no contexto atual.

4 METODOLOGIA

A distribuição do sistema operacional utilizado será o TurnKey Core em sua versão *11.1-lucid-x86*. Trata-se de uma distribuição Linux minimalista que tem como objetivo servir de base para servidores acessíveis de produção de *software* para *web*.

O *hardware* utilizado será:

- **Placa-Mãe**: ASUS P5GC-MX/1333
- **Processador**: Intel Core 2 Duo CPU E4600 @ 2.40GHz
- **Memória**: 2048MB RAM
- **Disco Rígido**: 30 GB
- **Placa de Vídeo**: NVIDIA GeForce 8600 GT

Alguns pacotes de *software* necessários aos testes serão obtidos através da ferramenta *apt* disponível no próprio TurnKey Core. São eles:

- **g++ e make**: Serão utilizados para compilar a distribuição do Node.js;
- **libssl-dev**: Pré-requisito para a compilação do Node.js;
- **apache2**: Versão pré-compilada do Apache HTTP Server;
- **PHP5**: Versão pré-compilada do PHP;
- **libapache2-mod-PHP5**: Módulo que integra o interpretador de código do PHP ao Apache HTTP Server.

Atualmente o Node.js não é disponibilizado em versões pré-compiladas, então o código fonte será obtido através da fonte recomendada pelo *site* oficial, e compilada em seguida.

O teste que será realizado através da ferramenta *ApacheBench* (*ab*) disponível na própria distribuição do Apache HTTP Server obtida. Essa ferramenta mede o desempenho de um servidor *web* através do envio de um dado número de requisições concorrentes.

Os parâmetros utilizados para os testes com a ferramenta ab foram:

- **-n20000**: Define que serão feitas 20.000 (vinte mil) requisições no total;
- **-c1000**: Define o nível de concorrência para 1.000 (mil) requisições ao mesmo tempo;
- **'http://127.0.0.1:80/'**: Define o endereço do servidor.

O uso de memória do sistema será medido através da ferramenta *free,* disponível no TurnKey Core. Essa ferramenta a quantidade de memória principal utilizada pelo sistema em intervalos determinados.

Os parâmetros utilizados foram:

- **-s0.5**: Define que a verificação será feita a cada 0,5 (meio) segundo;
- **-c60**: Define que serão feitas 60 (sessenta) medições totalizando trinta segundos.

O documento HTTP que será utilizado nas requisições de teste será semelhante para os dois servidores: Enviar a cadeia de caracteres *"Hello World"* (sem aspas) com poucas linhas de código.

O código utilizado nos testes com o Node.js está representado na Figura 1:

```
var http = require('http');

http.createServer(function (req, res) {
     res.writeHead(200, {'Content-Type': 'text/plain'});
     res.end('Hello World');
}).listen(80);
```

Figura 1 – Código de teste do Node.js

O código utilizado nos testes com o Apache HTTP Server e o PHP está representado na Figura 2:

```
<?php
echo "Hello World";
```

Figura 2 – Código do teste do Apache HTTP Server e do PHP

Os testes serão automatizados através de *Shell scripts*, que são arquivos que representam uma sequência de operações que devem ser enviadas ao interpretador de comandos do sistema operacional.

O *Shell Script* utilizado para o teste com o Node.js está representado na Figura 3:

```
#!/bin/bash
free -s0.5 -c60 > 'node.free.log' &
sleep 1s
node 'test.js' &
sleep 1s
ab -n20000 -c1000 'http://127.0.0.1:80/' > 'node.ab.log'
```

Figura 3 - Shell script para os testes com o Node.js

O *Shell Script* utilizado para o teste com o Apache HTTP Server e o PHP está representado na Figura 4:

```
#!/bin/bash
free -s0.5 -c60 > 'apache.free.log' &
sleep 1s
/etc/init.d/apache2 start &
sleep 1s
ab -n20000 -c1000 'http://127.0.0.1:80/' > 'apache.ab.log'
```

Figura 4 – Shell Script para os testes com o Apache HTTP Server e o PHP

A representação dos resultados dos testes será feita da seguinte maneira:

- **Testes de memória**: Os dados dos testes de memória representam todas as ocorrências da quantidade de memória principal em uso pelo sistema durante os testes. Esses dados vão exigir uma tabulação em planilha eletrônica e posteriormente a elaboração de um gráfico de comparação para que os resultados possam ser interpretados com mais facilidade e clareza.
- **Testes de desempenho**: A ferramenta *ab* já processa os dados dos testes de desempenho gerando um resumo simplificado. Esses resumos serão traduzidos e tabulados para facilitar a comparação.

5 DESENVOLVIMENTO

Os testes foram iniciados no dia 31 de maio de 2011 com a formatação do disco rígido e instalação do Turnkey Core na versão *11.1-lucid-x86*.

As dependências de compilação do Node.js foram adquiridas através da ferramenta *apt* nas seguintes versões, conforme lista obtida pelo comando *dpkg -l*:

- **make**: versão 3.81-7ubuntu1
- **g++**: versão 4:4.4.3-1ubuntu1

Foram obtidos pacotes utilizados na solução baseada em Apache HTTP Server e PHP nas seguintes versões:

- **apache2**: versão *2.2.14-5ubuntu8.4*
- **PHP5**: versão *5.3.2-1ubuntu4.9*
- **libapache2-mod-PHP5**: versão *5.3.2-1ubuntu4.9*

O código fonte do Node.js foi obtido através da ferramenta *git* conforme recomendação do site oficial do *software*. O comando utilizado está representado na Figura 5:

```
git clone --depth 1 https://github.com/joyent/node.git
```

Figura 5 – comando para obtenção do código fonte do Node.js

O processo de compilação foi executado através da seguinte sequência de comandos:

```
cd node
./configure
make
make install
```

Figura 6 - Comandos de compilação do Node.js

O Apache HTTP Server é configurado (durante a instalação) para ser inicializado automaticamente no sistema, mas como esse comportamento pode interferir nos resultados dos testes, ele foi desativado antes do início de ambos os testes.

Os códigos utilizados nos testes foram copiados para os diretórios de execução, concluindo o processo de preparação do ambiente de testes.

6 RESULTADOS

Os resultados dos testes foram obtidos nos seguintes arquivos, conforme configurado nos *shell scripts* usados nos testes:

- **node.free.log**: resultado dos testes de memória principal da ferramenta *free* para o Node.js
- **apache.free.log**: resultado dos testes de memória principal da ferramenta *free* para o Apache HTTP Server
- **node.ab.log**: resultado dos testes de desempenho da ferramenta *ab* para o Node.js
- **apache.ab.log**: resultado dos testes de desempenho da ferramenta *ab* para o Apache HTTP Server

Os resultados dos testes do uso de memória principal (através da ferramenta *free*) foram tabulados em planilha eletrônica, e foi elaborado gráfico comparativo descrito na Figura 7:

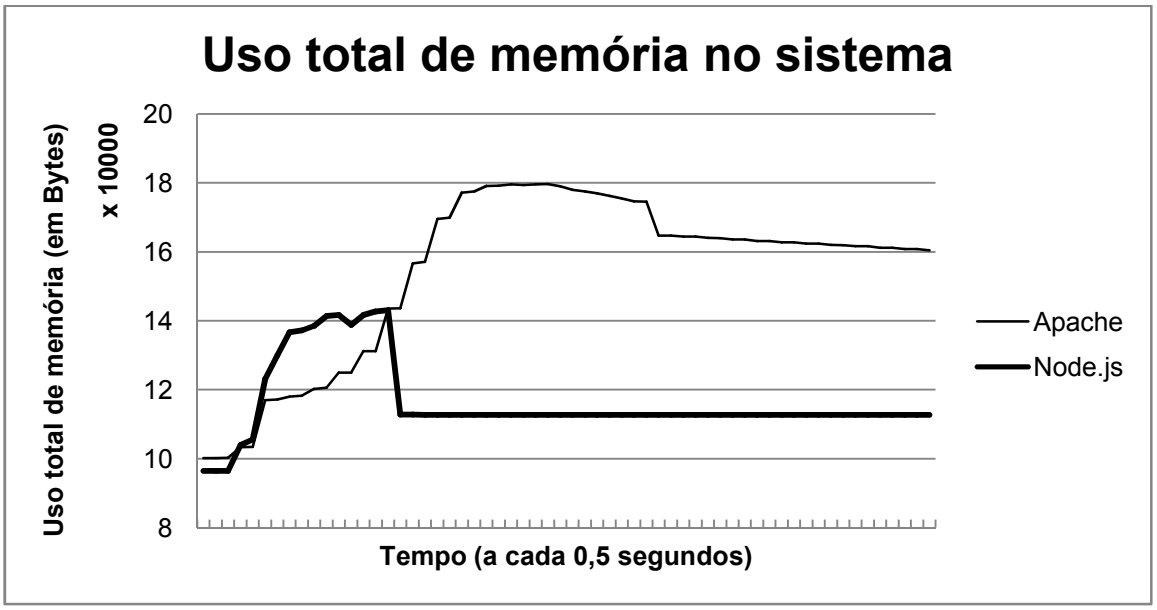

Figura 7 - Resultado dos testes de uso de memória principal

Os resultados dos testes de desempenho (através da ferramenta *ab*) foram tabulados e traduzidos, conforme descrito na Tabela 3:

Tabela 3 – Resultado dos testes de desempenho

	Node.js	Apache HTTP Server
Software do servidor	(Não informado)	Apache/2.2.14
Endereço do servidor	127.0.0.1	127.0.0.1
Porta do servidor	80	80
Caminho do arquivo	/	/
Tamanho do documento	11 bytes	11 bytes
Nível de concorrência	1000	1000
Tempo total do teste	5,835 s	15,970 s
Requisições efetuadas	20.000	20.000
Requisições que falharam	0	849
Total de dados transferidos	1.500.000 bytes	4.379.172 bytes
Dados HTML transferidos	220.000 bytes	216.986 bytes
Requisições por segundo (média)	3427,75	1252,38
Tempo por requisição (média)	0,292 ms	0,798 ms
Taxa de transferência	251,06 Kb/s	267,79 Kb/s

7 DISCUSSÃO E CONCLUSÃO

Devemos destacar que existem diferenças relevantes no funcionamento de cada um dos servidores utilizados nas soluções de teste, o que influenciou nos resultados dos testes.

Em relação ao desempenho, pudemos notar que o Node.js obteve resultados melhores em relação ao Apache HTTP Server, tanto no uso de memória principal quanto no número de requisições que esse pode suportar simultaneamente.

Em relação à programação, o código do Node.js implementa o servidor *web* como um método do módulo *"http"*. Esse método é usado para manipular as requisições manualmente, com poucas abstrações implícitas.

Já o Apache HTTP Server apresenta várias abstrações e otimizações por padrão. Isso agiliza o desenvolvimento do *software* na maioria dos casos, mas a implantação dessas características consome recursos do sistema.

Abstrações tornam a produtividade inversamente proporcional ao desempenho.

Em relação à disponibilidade, O Node.js é um software recente e dificilmente encontramos provedores de hospedagem *web* que tenham suporte a ele.

A combinação de Apache HTTP Server e PHP é mais comum do que o Node.js, sendo encontrada em boa parte dos provedores de hospedagem, e a preços acessíveis.

Em relação à documentação, O JavaScript (formalmente chamado de ECMAScript) é uma linguagem de programação padronizada pela ECMA e pela ISO. Essa linguagem é utilizada para a programação no Node.js.

Embora seja uma linguagem muito difundida por ser utilizada para programação *client-sided* em páginas da *web*, sempre foi criticada por implantar recursos problemáticos e com muitas exceções imprevisíveis.

Um exemplo muito conhecido descreve a dificuldade de se prever os resultados do operador de comparação fraca (==), pois o mesmo realiza conversões automáticas entre tipos de dados durante a comparação.

Como os documentos de padronização do ECMAScript são muito complexos e formais, a comunidade de programação cria suas próprias versões simplificadas de manuais para essa linguagem, incluindo exemplos e explicações adicionais. Tal material complementar pode ensinar técnicas de programação pouco

recomendáveis, acostumando programadores a conceitos que podem gerar problemas no futuro.

Em contrapartida, o PHP possui uma documentação oficial simplificada, mas que descreve as funcionalidades da linguagem de forma objetiva e exemplificada.

A linguagem PHP não é padronizada por nenhuma norma estrita e apresenta alguns problemas como:

- Diferenças de sintaxe entre versões do PHP;
- Diferenças entre formatos de nomes de funções;
- Diferenças na ordem de parâmetros de funções semelhantes;
- Depreciação e remoção de recursos que foram substituídos ou inutilizados em versões mais recentes do PHP;

Embora as duas linguagens tenham problemas específicos, a curva de aprendizado de ambas é muito baixa se comparada à de outras linguagens mais formais como C++ e Java.

Concluímos que o Node.js é uma escolha relevante quando se trata da resolução de problemas em que o desempenho é uma característica crítica da aplicação.

Utilizá-lo pode não ser interessante quando se trata de uma aplicação comum (em que o sistema não precisa ser otimizado com tanta intensidade), pois a falta de abstrações diminui a produtividade sem necessidade.

Para esse caso, a combinação do Apache HTTP Server com PHP é mais recomendada, pois ela se adapta a uma grande parte dos casos em que se precisa criar aplicações para *web*.

Também não podemos descartar a possibilidade de se integrar as duas soluções, utilizando-as em conjunto e delegando operações adequadas às características de cada tecnologia.

8 REFERÊNCIAS

Flórida: Wikimedia Foundation, **Node.js.** <http://en.wikipedia.org/wiki/Nodejs>, acesso em 9 de mai de 2011.

HIRATA, R., **Otimizando Servidores Web de Alta Demanda.** 2002. 202 p. Mestrado em Ciência da Computação – UNICAMP, Campinas, 2002.

Internet World Stats, **World Internet Usage and Population Statistics.** Disponível em: <http://www.internetworldstats.com/stats.htm>, acesso em 20 de abr de 2011.

KEGEL, D., **The C10k Problem.** Disponível em: <http://www.kegel.com/c10k.html>, acesso em 20 de abr de 2011.

LEITE, E. M., **Introduçao ao Node.js**. In: CAMPUS PARTY, 4, 2011,São Paulo. Disponível em: <http://www.youtube.com/watch?v=XPjogYLuFNg>, acesso em 2 de mai de 2011.

PHP Documentation Group, **Manual do PHP.** Disponível em: <http://br.php.net/manual/pt_BR/preface.php>, acesso em 9 de mai de 2011.

The Apache Software Foundation, **Apache HTTP Server.** Disponível em: <http://projects.apache.org/projects/http_server.html>, acesso em 9 de mai de 2011.

TIOBE Software, **TIOBE Programming Community Index for May 2011.** Disponível em: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>, acesso em 2 de mai de 2011.

W3Techs, **Usage of web servers for websites.** Disponível em: <http://w3techs.com/technologies/overview/web_server/all>, acesso em 9 de mai de 2011.