

O melhor servidor Web do mundo

Forte Apache

O Apache é o servidor Web mais usado em máquinas Linux. Extremamente poderoso e flexível, conheça em profundidade o "A" do "LAMP".

por Jorge Pereira



Sarah Williams – www.sxc.hu

Muitas pessoas que trabalham com informática, principalmente na área de redes, já devem ter ouvido falar algo sobre o Apache, um software muito conhecido, presente em praticamente todas as distribuições GNU/Linux e cotado como um dos softwares mais respeitados no mundo. Após mostrar a gênese do Apache, este artigo aborda também o processo de instalação do servidor e estuda a criação de uma configuração inicial do Apache, suficiente para torná-lo operacional em conjunto com os demais componentes de uma solução LAMP (Linux, Apache, MySQL, PHP).

O servidor Apache

O Apache Server – oficialmente chamado HTTP Server Project, já que existe uma infinidade de outros produtos com a chancela Apache – iniciou seus trabalhos de forma oficial em 1995, quando foi lançada a versão inicial 0.6.2. A última versão estável é a 2.2.3, que desponta, segundo os próprios desenvolvedores, para uma nova versão, mais estável e com novas funcionalidades, do Apache. Paralelamente, a série anteriormente considerada estável do Apache ganhou uma nova versão legacy, ou seja, de compa-

tabilidade com sistemas anteriores. É a versão 2.0.59, que apresenta alguns patches de segurança para o módulo `mod_rewrite`.

A conceituada empresa inglesa Netcraft [1], responsável por realizar periodicamente pesquisas sobre servidores Web, constatou, em sua última pesquisa, que mais de 60% dos servidores utiliza mundialmente o Apache como software. Essa preferência pode ser tratada, informalmente, como uma garantia que comprova a qualidade e segurança anexos ao projeto (quadro 1).

Quadro 1: Pesquisa Netcraft entre agosto/1995 e julho/2006

Empresa	Junho/2006	Percentual	Julho/2006	Percentual	Mudanças
Apache	52389885	61.25	55622584	63.09	1.84
Microsoft	25415611	29.71	25988099	29.48	-0.23
Zeus	531399	0.62	518503	0.59	-0.03
Sun	1311822	1.53	347037	0.39	-1.14

Canivete suíço

O servidor Apache é capaz de executar scripts nas linguagens *PHP*, *Perl*, *TCL*, *Python*, *CGI* (quadro 2) e até mesmo *ASP* (embora com suporte deficiente), podendo atuar como servidor *FTP*, *HTTP*, entre outros. Sua utilização mais conhecida é a que combina o Servidor Apache com a linguagem de programação para Web *PHP* e o banco de dados *MySQL* ou *PostgreSQL*.

As versões 1.x e 2.x do Apache possuem diferentes arquiteturas. A mais recente oferece muitos aperfeiçoamentos em relação às versões anteriores. Os recursos e mudanças mais importantes são listados nos próximos parágrafos.

Módulos de multi-processamento

A primeira mudança importante no Apache 2.0 é a introdução dos módulos de multiprocessamento (*MPMs*, ou *Multi Processing Modules*). Para entender por que os *MPMs* são criados, é preciso compreender como o Apache funcionava antes, pois nas versões 1.3 e anteriores utilizava-se uma arquitetura de *preforking*. Nessa arquitetura, o processo pai do Apache sofria *forks* para um conjunto de processos filhos, cada um atendendo a uma das solicitações que chegavam. O processo pai simplesmente monitorava os filhos e criava ou eliminava processos filhos de acordo com a quantidade de solicitações recebidas. Após muitos estudos e testes desse modelo, concluiu-se que ele não oferecia um desempenho aceitável em sistemas não concentrados em processo, como é o caso do *Windows*®. Dessa forma, a utilização dos *MPMs* passa a ser responsável por iniciar os processos servidores e por atender à solicitação por meio de processos filhos ou *threads*, dependendo da implementação do *MPM*. Vários *MPMs* estão disponíveis, alguns deles serão descritos abaixo.

Prefork

O *MPM* de *prefork* reproduz a arquitetura utilizada nas versões do Apache 1.3 e anteriores, nas quais era criado todo um conjunto de processos filhos para atender às solicitações. Com o *prefork*, cada processo filho obtém uma única *thread*. Vejamos o exemplo: se iniciarmos 30 processos filhos, o Apache poderá atender às 30 solicitações simulta-

neamente. Caso ocorra algum erro e o processo venha a morrer ou receber um sinal *SIGTERM* ou *SIGKILL*, somente uma solicitação se perderá, pois o número de processos filhos é controlado pelo arquivo de configuração. Quando o número de requisições aumentar, os novos processos filhos serão acrescentados, até ser alcançado o número máximo. De modo semelhante, quando as requisições diminuem, todos os processos filhos extras são eliminados.

MPM threaded

Esse *MPM* é responsável pelo suporte a *threads* na série 2.0 do Apache, e é semelhante ao *MPM Prefork*. Porém, em vez de ter uma única *thread*, cada processo filho pode ter um número especificado de *threads*. Cada processo filho pode ter um número predeterminado de *threads*, e cada *thread* dentro de um processo filho pode atender a uma solicitação diferente. No caso, se o Apache iniciar 30 processos filhos e cada filho tiver permissão para ter no máximo 10 *threads*, então o Apache poderá atender a $30 \times 10 = 300$ requisições simultâneas. Por exemplo, se um módulo experimental fizer a *thread* receber um sinal para morrer, o processo inteiro morrerá. Isso significa que todas as solicitações que estiverem sendo atendidas pelas *threads*

dentro daquele mesmo processo filho serão perdidas. Devido ao fato de as solicitações se distribuírem entre as *threads* em processos filhos separados, a morte de um deles somente causa a queda das conexões atendidas pelo mesmo processo filho. Considerando que *threads* são mais eficientes do que processos no uso de recursos do sistema, esse *MPM* contribui muito para a escalabilidade.

MPM per child

Essa é uma novidade controversa da série 2.0, pois esse módulo *MPM*, inicia um número pré-definido de processos filhos com um número predeterminado de *threads*. À medida que a carga de requisições aumenta, os processos abrem novas *threads*, conforme necessário. Quando a contagem de requisições reduz, os processos condensam sua contagem de *threads*, empregando uma configuração de contagem mínima e máxima de *threads*. A diferença fundamental entre esse módulo e o *MPM threaded* é que, nele, o número de processos é estático, e cada processo pode funcionar utilizando um *UID* ou *GID* diferente, o que facilita a execução de múltiplos websites virtuais. O uso do módulo *per child* aumenta o nível de segurança, pois limita o acesso a determinados pontos do servidor. ➤

Quadro 2: Recursos do Apache

- Possui suporte a scripts *CGI* utilizando linguagens como *Perl*, *PHP*, *Shell Script*, *ASP* etc
- Suporte à autorização de acesso, podendo ser especificadas restrições para cada endereço, arquivo, diretório ou grupo de usuários acessado no servidor
- Autenticação por nome de usuário e senha para acesso a alguma página, subdiretório ou arquivo (com suporte a criptografia via *Crypt* e *MD5*), e autenticação em banco de dados
- Negociação de conteúdo, permitindo a exibição da página Web no idioma requisitado pelo navegador cliente
- Suporte a tipos *MIME*
- Suporte a *IP virtual hosting*
- Suporte a *name virtual hosting*
- Personalização de logs e mensagens de erro
- Suporte a *virtual hosting*, sendo possível servir duas ou mais páginas, com endereços e portas diferentes, através do mesmo processo, ou usar mais de um processo para controlar mais de um endereço
- Suporte a servidor proxy *FTP* e *HTTP*, com limite de acesso e *caching* (todos flexivelmente configuráveis)
- Suporte a proxy e redirecionamentos baseados em URLs para endereços internos
- Suporte a criptografia via *SSL* e certificados digitais
- Por ser um software modular, pode ser personalizado de forma a utilizar menos recursos da máquina

MPM WinNT

Esse é o MPM para a plataforma Windows, incluindo o Windows 2000, Windows NT e Windows 9x. É um módulo multithreaded. Com esse módulo, o Apache criará um processo pai e um processo filho, e o processo filho gerará todos os threads que atenderão às requisições feitas ao servidor. Além disso, agora esse módulo também aproveita algumas chamadas de funções nativas do Windows. Dessa forma, é possível que ele funcione melhor que as versões anteriores do servidor Apache desenvolvidas para essa plataforma.

Filtragem de entrada e saída

O Apache 2.0 inclui a arquitetura de E/S em camadas. Isso significa que a saída de um módulo pode se tornar a entrada de outro. Esse efeito de filtragem é muito interessante. Por exemplo, a saída produzida por scripts CGI é processada pelo módulo *mod_cgi*, e pode então ser entregue ao módulo *mod_include* responsável pelas SSI (*Server Side Includes*). Em outras palavras, os scripts CGI podem produzir a

saída sob a forma de etiquetas de SSI, que podem ser processadas antes da saída do conteúdo final para a Web. Segundo os próprios desenvolvedores do Apache, diversas outras aplicações da filtragem de E/S estarão disponíveis no futuro.

APR (Apache Portable Runtime)

Como se não bastasse a forma como o Apache Group desenvolveu o servidor Web mais popular no mundo, a versão 2.0 trouxe a necessidade de cuidar da sua portabilidade. Nas versões anteriores, o Apache lidava internamente com a portabilidade, o que tornava a base de código menos flexível, dificultando um pouco o seu entendimento e manutenção. Pensando nisso, o Apache Group introduziu o APR (*Apache Portable Runtime*). A finalidade do APR é oferecer uma única interface, em linguagem C, para funções específicas da plataforma, de tal forma que o código possa ser escrito uma única vez, aumentando o nível de desenvolvimento de novas aplicações com uma mesma API. Isso torna o Apache capaz

de funcionar de forma mais fluida e natural em plataformas diversas, como Windows, BeOS, Amiga e OS/2. Graças ao APR, programas como o *ApacheBench* [2] podem ser utilizados em todas as plataformas.

Mão na massa

Agora que entramos um pouco mais na arquitetura e funcionamento do servidor Apache, vamos instalar nossa primeira versão a partir do código-fonte. Nessa tarefa, naturalmente são necessários o compilador GCC, as ferramentas *Autotools* e a biblioteca *libc6*. Com os comandos a seguir, realizamos a configuração, compilação e instalação do Apache.

```
# wget -c http://www.apache.org/dist/
# httpd/httpd-2.0.59.tar.gz
# tar -xvzf httpd-2.0.59.tar.gz
# cd httpd-2.0.59/
# ./configure
# --sysconfdir=/etc/apache2 \
# --prefix=/usr/local \
# --localstatedir=/var/log/apache2 \
# --datadir=/dominios/default
# make
# make install
```

Note que, de acordo com nossa instalação acima, todas as configurações do nosso servidor ficarão em */etc/apache2*, os registros de logs em */var/log/apache2*, e os binários executáveis em */usr/local/bin*. Os dados com o conteúdo de nosso domínio padrão serão armazenados em */dominios/default*, enquanto os dos outros domínios adicionais naturalmente serão depositados em */dominios/outro_dominio*. Para mais informações sobre os parâmetros de compilação, basta abrir o arquivo *INSTALL* dentro do diretório dos fontes do Apache.

Pontapé inicial

Nosso servidor possui um aplicativo chamado *apachectl*, que é utilizado para a manipulação do *daemon httpd*. Vamos ini-

Exemplo 1: Configurações globais do Apache

```
01 IP ou host padrão: neste caso vamos utilizar uma configuração para responder em todas as interfaces de rede
02 ServerName *:80
03
04 # E-mail do administrador do servidor
05 ServerAdmin sys-admin@meuhost.com.br
06
07 # Diretório raiz de conteúdo
08 DocumentRoot /dominios/default/htdocs
09
10 # Configurando os Nomes dos Arquivos Padrão
11 DirectoryIndex index.html main.html index.cgi
12
13 # Acesso apenas aos diretórios devidos
14 <Directory "/dominios/*/htdocs">
15 Options Indexes FollowSymLinks
16 AllowOverride None
17 </Directory>
18
19 # Acesso para execução de scripts CGI
20 <Directory "/dominios/*/cgi-bin">
21 Options -Indexes +ExecCGI
22 AllowOverride None
23 </Directory>
24
25 # Configurando os Nomes dos Arquivos Padrão a serem exibidos antes de ser listado o conteúdo de um diretório
26 DirectoryIndex index.html index.html index.cgi
27
28 # Diretório com os scripts CGI
29 ScriptAlias "/cgi-bin/" /dominios/default/cgi-bin/
30
31 # Habilitando execução de scripts CGI
32 AddHandler cgi-script .cgi
```

Exemplo 2: Configurações para domínios virtuais

```
01 # Caso queira que o Apache responda por todas as requisições feitas às todas
    ↳ interfaces de rede, basta seguir este exemplo
02 NameVirtualHost *:80
03
04 # Primeiramente, devemos ter uma configuração padrão para os domínios virtuais,
    ↳ para ser exibido o conteúdo quando o host solicitado não existir no servidor.
05
06 <VirtualHost *:80>
07 ServerAdmin sys-admin@meuhost.com.br
08 DocumentRoot /domínios/default/htdocs
09 ServerName localhost
10 </VirtualHost>
```

cializar o serviço por meio do comando `/usr/local/bin/apachectl start`. Caso apareça alguma mensagem de erro, verifique nos arquivos de log em `/var/log/apache2`. Se tudo ocorreu tranqüilamente, basta acessar a página inicial do Apache no seu navegador, utilizando o endereço IP da máquina local, ou utilizando o endereço <http://localhost>.

Antes de começarmos a configuração, devemos sempre definir a forma de armazenamento dos websites hospedados pelo nosso servidor. Em nosso exemplo, aprenderemos a configurar um domínio virtual, e utilizaremos como ponto de armazenamento o diretório `/domínios`. Então, os arquivos do domínio exemplo serão armazenados em `/domínios/exemplo`, com os documentos HTML ficando no subdiretório `htdocs` e os scripts CGI em `cgi-bin`.

Supondo que tenhamos o domínio `meuhost.com.br` já registrado num servidor DNS, e que queiramos hospedá-lo em nosso servidor recém-instalado, vamos preparar os diretórios para armazenar o website. Mas primeiro, vamos aprender um pouco sobre as configurações disponíveis no arquivo principal, `/etc/apache2/httpd.conf`. Abra esse arquivo com seu editor de texto predileto e, caso as configurações já existam, basta ir comentando ou descomentando as diversas opções, de acordo com os exemplos 1 e 2.

Para adicionar um domínio virtual ao nosso servidor, devemos primeiramente criar o diretório a partir de `/domínios` e, logo em seguida, criar as entradas de configuração no `httpd.conf`. O primeiro passo é criar os diretórios:

```
# mkdir -p /domínios/meuhost.com.br/htdocs
# mkdir -p /domínios/meuhost.com.br/
↳ cgi-bin
```

Depois, podemos alterar o arquivo `/etc/apache2/httpd.conf`:

```
<VirtualHost *:80>
ServerAdmin webmaster@meuhost.com.br
DocumentRoot /domínios/meuhost.com.br/
↳ htdocs/
ServerName www.meuhost.com.br
ScriptAlias "/cgi-bin/"
↳ /domínios/meuhost.com.br/cgi-bin/
</VirtualHost>
```

Feito isso, basta agora parar e iniciar o servidor Apache. Caso não funcione, procure refazer todo o processo, e lembre-se de que qualquer tipo de erro será relatado pelo Apache no arquivo `/var/log/apache2/logs/error_log`. Funcionando, reinicie o servidor com `/usr/local/bin/apachectl restart`.

Teste o acesso através de um navegador Web e informe o domínio configurado (em nosso exemplo, `www.meuhost.com.br`). Todo o conteúdo dentro de `/domínios/meuhost.com.br/htdocs/` será exibido. Caso você informe qualquer outro domínio que esteja direcionado para seu IP e que não esteja cadastrado no Apache como um "Domínio Virtual", ele exibirá o conteúdo do diretório `/domínios/default/htdocs`. ■

Mais Informações

- [1] Pesquisa realizada em 2005 pela Netcraft: http://news.netcraft.com/archives/2005/12/02/december_2005_Web_server_survey.html
- [2] ApacheBench: <http://freshmeat.net/projects/apachebench/>
- [3] W3C HTML: <http://www.w3.org/TR/REC-html40/>
- [4] W3C CGI: <http://www.w3.org/CGI/>
- [5] CGI utilizando scripts de shell: http://www.thobias.org/doc/cgi_shell.html

Instant Messaging

**Não proíba!
Gerencie e Controle!**

Com o messengerPOLICY®, você conta com todas as facilidades de que precisa para garantir o uso produtivo e seguro do MSN® em sua empresa.

- Interface web de administração
- Orientado a objetos (usuários, grupos, horários e recursos)
- Uso personalizado do MSN® conforme suas necessidades
- Monitoramento on-line das mensagens trocadas
- Histórico de todas as conversas
- Gráficos estatísticos para gerenciamento

No messengerPOLICY®, o administrador pode criar os mais diversos tipos de regras, como por exemplo:

JOÃO (OBJETO - USUÁRIO) PODE CONVERSAR COM COLEGAS (OBJETO - GRUPO DE USUÁRIOS) RESPEITANDO LAZER (OBJETO-TABELA DE HORÁRIO)



**messenger
POLICY**

**Não seja alvo de pragas virtuais,
proteja-se com o messengerPOLICY®,
saiba mais acessando www.brc.com.br.**



BRCONNECTION
CRESCER SEGURO

www.brc.com.br | 55 (11) 2165-8888