Escola Superior Aberta do Brasil
Lato Sensu em Engenharia de Sistemas

COMPARAÇÃO ENTRE SERVIDORES WEB APACHE HTTP SERVER E NGINX

Victor Costa de Alemão Cisneiros[1]
Hudson Ramos[2]

Resumo

Neste trabalho, será realizada uma comparação entre o Apache HTTP Server, servidor web de mais de 20 anos e atual líder de mercado, e o Nginx, o novo servidor web que tem sido adotado cada vez mais por diversos administradores de sistemas no mundo todo. Será verificado se os dois softwares são capazes de atender a demanda exigida pelos sistemas de informações atuais. Para isso, será feita uma comparação entre as funcionalidades suportadas por cada um desses softwares, além de uma série de testes experimentais, com o objetivo de investigar o desempenho deles sob uma alta carga de acessos simultâneos. Ao final do artigo, será apresentado, baseando-se nos testes realizados, qual desses dois servidores oferece melhor desempenho.

Palavras-chave: Servidores Web. Nginx. Apache HTTP Server. Comparação. Desempenho. Benchmark.

[1] Pós-graduando em Engenharia de Sistemas na Escola Superior Aberta do Brasil – ESAB.
E-mail: victorcisneiros@gmail.com
[2] Mestre em Engenharia de Software – UFES.

1. Introdução

Com a popularização da web e as mídias sociais, tem se tornado cada vez mais necessário a utilização de servidores robustos para atender a todo o tráfego gerado na internet. Esse fato tem se agravado com a recente popularização dos dispositivos móveis, como os smartphones e tablets. Cada vez mais pessoas estão acessando a internet, consumindo maiores quantidades de informações por mais tempo.

Desde o início da World Wide Web, o Apache HTTP Server tem sido o principal servidor web, dominando o mercado até os dias de hoje (NEDELCU, 2013). A necessidade de suportar maiores quantidades de tráfego tem feito, porém, surgir uma série de novos servidores web, dentre os quais destaca-se o Nginx, que já é hoje o segundo servidor web mais utilizado na internet.

O surgimento de novos softwares levanta alguns questionamentos sobre se ele é adequado para o uso em grandes projetos comerciais. O objetivo deste trabalho será realizar um comparativo entre as funcionalidades e o desempenho dos dois servidores web mais utilizados em 2015, o Apache HTTP Server e o Nginx, de modo a determinar a viabilidade deles em empresas e projetos importantes. A comparação irá focar em observar o quão bem estes servidores suportam manter seu desempenho ideal quando sob uma alta carga de acessos simultâneos.

A metodologia adotada para este artigo será uma pesquisa exploratória com coleta de dados por meio de pesquisa bibliográfica e experimental. O experimento será realizado por meio de testes com os softwares Apache HTTP Server e Nginx, no ambiente Linux. Os resultados serão apresentados na forma de quadro comparativo, gráficos e tabelas.

2. Servidores Web

O termo servidor web pode se referir tanto ao hardware quanto ao software que sabe comunicar-se com os clientes utilizando o protocolo Hypertext Transfer Protocol (HTTP) (ERIC LARSON, 1999). Neste artigo, o termo servidor web será usado para se referir especificamente ao software.

A função principal de um servidor web é a de entregar documentos aos computadores e dispositivos clientes. Estes documentos usualmente consistem em páginas HTML, que podem incluir imagens, folhas de estilo e scripts, além do seu conteúdo texto (DALZIEL, 2015).

O primeiro servidor web, o CERN httpd, foi criado em 1990 pelos programadores Tim Berners-Lee, Ari Luotenen e Henrik Frystik Nielsen na Organização Europeia para a Pesquisa Nuclear (CERN). Foi originalmente desenvolvido em C para o sistema operacional NeXTSTEP (INTERNET GUIDE, 2015).

Anos mais tarde, no Centro Nacional de Aplicações de Supercomputação (NCSA), foi lançado um novo servidor web, chamado NCSA httpd. O servidor da NSCA rapidamente se tornou o servidor web mais popular entre os anos 1993 e 1995, entretanto, seu desenvolvimento foi cancelado. O Apache, objeto de estudo desse artigo, surgiu tendo como base o código deste servidor da NCSA (ERIC LARSON, 1999).

3. Apache HTTP Server

O projeto Apache HTTP Server é um esforço de desenvolvimento de software colaborativo com o objetivo de criar um servidor HTTP robusto, com muitas funcionalidades, com qualidade comercial e com todo o seu código-fonte disponível gratuitamente (APACHE SOFTWARE FOUNDATION, 2015). O projeto faz parte da Apache Software Foundation, uma das maiores comunidades colaborativas de programadores na internet, focada no desenvolvimento de soluções de software gratuitas e de código aberto.

O desenvolvimento do Apache teve como base o código do NCSA httpd versão 1.3. Uma série de melhorias e correções de bugs foram feitas neste código e lançadas como o servidor Apache em abril de 1995.

O lançamento do servidor Apache foi um grande sucesso, em menos de 1 ano depois do lançamento dele, o Apache já se tornara o servidor web mais popular da internet, mantendo essa posição até os dias de hoje (W3TECHS, 2015).

O Apache é conhecido por fazer parte da famosa pilha de softwares conhecida como LAMP (Linux, Apache, MySQL e PHP), um conjunto de programas de código aberto que formam uma poderosa plataforma para o desenvolvimento de aplicações web. A pilha LAMP é incluída em praticamente quase todas as distribuições Linux (WEBOPEDIA, 2015).

4. Nginx

O Nginx é um novo servidor web, lançado publicamente em 2004. Ele foi desenvolvido pelo programador russo Igor Sysoev, que havia escrito em 2001, um módulo para o Apache chamado mod_accel, como um substituto melhor para o módulo mod_proxy. Segundo Igor, ficou claro durante o desenvolvimento daquele módulo, que o Apache possuía uma escalabilidade baixa (SYSOEV, 2008).

Havia nessa época, uma discussão na internet sobre como escalonar um servidor para suportar 10.000 conexões simultâneas. A web era um grande sucesso e já era a hora de os servidores conseguirem suportar tamanha carga (KEGEL, 1999). Essa discussão ficou conhecida como o C10K Problem. Igor Sysoev leu ela, estudou os servidores existentes e decidiu que queria um software parecido com o Apache, mas que além de funcionalidades como Server Side Includes (SSI), proxy e suporte a cache, possuísse também uma escalabilidade alta.

Assim nasceu o servidor Nginx, que logo passou a ser adotado por alguns dos sites mais populares da Rússia, como o portal Rambler, que em 2008 já servia mais de 500 milhões de requisições por dia usando o Nginx como servidor (NEDELCU, 2013).

O Nginx tem ganhado cada vez mais adeptos ao longo dos anos. O site W3Techs, especializado em dados estatísticos de tecnologias da web, informou que em julho de 2015, o Nginx já era o segundo servidor web mais utilizado na Internet, ocupando inclusive, a primeira posição, quando restringido os dados a apenas os 1.000 sites mais visitados de acordo com o ranking da Alexa (W3TECHS, 2015).

5. Comparação entre Apache e Nginx

Características Principais

Esta seção mostra as principais características e diferenças entre os servidores Apache e Nginx:

Funcionalidade	Nginx	Apache
Gerenciamento de Requisições: Especifica como o servidor processa cada requisição do cliente	**Arquitetura orientada a eventos**: Nesse tipo de arquitetura, são utilizados sockets assíncronos e as requisições não são processadas por threads separadas. O objetivo é reduzir o gasto de memória e CPU.	**Sockets, Threads e Processos síncronos**: Nesta arquitetura, cada requisição está em uma thread ou processo separado, que utiliza sockets síncronos.
Linguagem de Programação: Especifica a linguagem em que o software do servidor foi escrito.	**C**: A linguagem C é notavelmente de baixo nível e oferece recursos de gerenciamento de memória mais exatos.	**C**: Apesar do Apache ter sido escrito em C, vários dos seus módulos foram feitos em C++.
Portabilidade: Especifica quais sistemas operacionais são suportados pelo servidor.	**Multiplataforma**: O Nginx suporta os sistemas operacionais Windows, GNU/Linux, Unix, BSD, Mac OS X e Solaris.	**Multiplataforma**: O Apache suporta os sistemas operacionais Windows, GNU/Linux, Unix, BSD, Mac OS X, Solaris, Novell NetWare, OS/2, TPF, OpenVMS, eCS, AIX, z/OS, HP-UX e mais.
Ano de nascimento: Especifica o ano em que o desenvolvimento começou.	**2002**: Sendo mais novo que o Apache, o Nginx foi criado para uma era mais moderna em que os servidores necessitavam de melhor desempenho e concorrência.	**1994**: O Apache é um dos numerosos projetos de código aberto iniciados na década de 90, que contribuíram para fazer da World Wide Web o que ela é hoje.
HTTPS: Especifica se o servidor web suporta ou não entregar páginas web seguras utilizando o protocolo HTTPS.	**Suporte através de módulo**: O Nginx possui um módulo HTTPS que é incluído por padrão durante a compilação.	**Suportado através de módulo**: O Apache possui um módulo HTTPS que é incluído por padrão na instalação.
Virtual Hosting: Funcionalidade que permite	**Suportado Nativamente**: O Nginx possui suporte a	**Suportado Nativamente**: O Apache possuir suporte a

um servidor web hospedar múltiplos sites em uma mesma máquina.	Virtual Hosting mas não está configurado por padrão para aceitar um arquivo de configuração distinto para cada Virtual Host.	Virtual Hosting e permite a inclusão de um arquivo de configuração distinto (.htaccess) por diretório.
Suporte a CGI: Especifica os protocolos CGI que o servidor suporta, para a geração de conteúdo dinâmico.	**FastCGI, WSCGI, SCGI**: O Nginx suporta esses 3 protocolos CGI através de módulos, que devem ser incluídos durante a compilação.	**CGI, FastCGI**: O Apache suporta a maioria dos protocolos CGI através de módulos que podem ser carregados em um arquivo de configuração.
Sistema de Módulos: Especifica se o servidor suporta a extensão de suas funcionalidades através de um sistema de módulos.	**Sistema de Módulos Estático**: O Nginx possui um sistema de módulos estático. Cada módulo deve ser habilitado ou desabilitado durante a compilação do programa.	**Sistema de Módulos Dinâmico**: O Apache possui um sistema de módulos dinâmico, permitindo que os módulos sejam habilitados e desabilitados por meio de arquivos de configuração.

Quadro 1: Comparação de características entre o Nginx e Apache
Fonte: (NEDELCU, 2013)

Comunidade

Algumas das principais questões que devem ser respondidas ao se adotar um projeto de código aberto são (NEDELCU, 2013):

- Onde vou conseguir suporte ao encontrar um problema?
- Vou encontrar documentação sobre as funcionalidades oferecidas pelo software?
- Irão mais módulos serem implementados no futuro?
- O projeto está ativo e sendo atualizado pelos seus desenvolvedores?
- A segurança do software foi testada por uma equipe grande o suficiente de administradores?

Não há dúvida de que o Apache atende a essas questões. Lançado em 1995, ele rapidamente se tornou o servidor HTTP mais popular da internet (APACHE SOFTWARE FOUNDATION, 2015) e passou a ser incluído e suportado pelas principais distribuições Linux, tais como a Red Hat Enterprise Linux e o Debian. A página oficial de módulos do Apache lista um total de mais de 500 módulos oferecendo diversas funcionalidades adicionais (NEDELCU, 2013).

Além de tudo isso, o Apache faz parte da Apache Software Foundation, uma das maiores comunidades colaborativas de desenvolvedores da internet, com mais de 2.600 voluntários.

Seus projetos são usados por alguma das maiores empresas de computação do mundo como o Facebook e o Google, e incluem softwares famosos como o Apache HTTP Server, Cassandra, Hadoop, OpenOffice, Maven, Subversion, Tomcat, SpamAssassin, entre outros (APACHE SOFTWARE FOUNDATION, 2015).

Lançado publicamente em 2004, o Nginx tem sido atualizado constantemente e não há indícios de que o desenvolvimento irá parar, pois foi adotado por boa parte dos maiores sites da internet (W3TECHS, 2015). A wiki oficial do projeto lista 56 módulos oficiais e 112 módulos feitos por terceiros. Esses módulos incluem funcionalidades modernas como suporte ao protocolo SPDY, configuração utilizando-se a linguagem Javascript (KRILL, 2015) e suporte à nova versão 2 do protocolo HTTP (NGINX INC., 2015).

Em 2013, a Nginx Inc., empresa responsável pelo desenvolvimento do Nginx, anunciou uma versão comercial do seu servidor web, chamada de Nginx Plus. Essa nova versão oferece algumas funcionalidades extras como balanceamento de carga avançado, reconfiguração dinâmica e monitoramento de atividades, além de um serviço de suporte para configuração e otimização do software (NGINX INC., 2013). O lançamento do Nginx Plus pode dar um conforto maior para as empresas que gostariam de adotar o Nginx, mas não querem depender apenas do suporte de um projeto gratuito e de código aberto.

Desempenho

Um dos principais fatores que afetam o desempenho do Apache é a escolha do módulo de gerenciamento de requisições. A versão 2.4 para GNU/Linux oferece 3 módulos distintos:

- **mpm_prefork**: Neste módulo, o Apache implementa uma abordagem sem threads, onde um processo pai gerencia um pool de processos filhos que são capazes, cada um, de atender 1 requisição simultânea. É o melhor gerenciador para isolar cada requisição, sendo apropriado, portanto, quando se deseja utilizar bibliotecas que não suportam threads (APACHE SOFTWARE FOUNDATION, 2015).
- **mpm_worker**: Neste módulo, o Apache implementa uma abordagem híbrida multi-processo / multi-thread. Cada requisição é atendida por uma thread distinta. O objetivo é economizar memória em relação ao prefork, visto que uma thread consome menos recursos do que um processo novo (APACHE SOFTWARE FOUNDATION, 2015).
- **mpm_event**: Este módulo é um caso especial do mpm_worker, desenvolvido para permitir que mais requisições simultâneas possam ser atendidas ao mesmo tempo.

> Isso é feito delegando parte do trabalho à threads de suporte, liberando assim as threads principais para já começarem a atender as novas requisições. O objetivo é resolver o problema das conexões persistentes (keep-alive), em que o cliente pode manter a conexão aberta depois da primeira requisição, e usar ela para as futuras requisições (APACHE SOFTWARE FOUNDATION, 2015).

Ao contrário do Apache, o Nginx não cria um novo processo ou thread para cada nova conexão. Ao invés disso, ele utiliza uma arquitetura de programação orientada a eventos, permitindo com que cada processo possa atender várias conexões ao mesmo tempo, possibilitando assim com que mesmo uma máquina razoável possa suportar uma grande carga de requisições simultâneas (AMY BROWN, 2012).

Criar processos ou threads separadas requerem um novo contexto de execução, incluindo alocação de memória na heap e na stack, além disso, tempo precioso de CPU é gasto para criar esses itens e também na constante troca de contextos. Todas essas complicações fazem com que arquiteturas antigas como o do Apache não sejam adequadas para aguentar um número muito alto de conexões (AMY BROWN, 2012).

Na arquitetura do Nginx, os processos, que são chamados de workers, aceitam novas requisições ouvindo um socket compartilhado e então executando um loop que processa milhares de conexões ao mesmo tempo. Não há nenhuma distribuição de conexões aos workers no software, tudo é feito utilizando-se as chamadas assíncronas e orientada a eventos dos sistemas operacionais, como o epoll do Linux. A ideia é bloquear a execução o menos possível para que um só processo consiga atender várias conexões ao mesmo tempo (AMY BROWN, 2012).

Uso

De acordo com os dados mais recentes da W3Techs (W3TECHS, 2015), o Apache HTTP Server é o servidor web mais utilizado na internet, com uma fatia de 57% do mercado. O Nginx aparece em segundo lugar com 24,8%, seguido pelo Microsoft Internet Information Services com 13,1%.

Um dado interessante a se notar é que a W3Techs também oferece informações quando restringido os dados somente aos sites mais visitados segundo o ranking da Alexa. Quando se considera apenas os 1.000 sites mais visitados, o Nginx passa a ser o servidor mais popular, com uma fatia de 43,5% do mercado, contra 28,3% do Apache. A liderança do Nginx se mantém

mesmo quando se considera apenas os 10.000 sites mais populares, com o Nginx sendo usado por 46% destes sites contra 32,8% do Apache (W3TECHS, 2015).

Outro dado importante é notar o crescimento do Nginx nos últimos anos. Em janeiro de 2010 ele possuía somente 3,9% de mercado, mas esse número cresceu para 14,1% em 2013 e 22,9% em janeiro de 2015 (NGINX INC., 2015).

6. Testes

Nesta seção, serão detalhados os testes realizados para comparar o desempenho entre os servidores Apache e Nginx. Os experimentos consistem em realizar um alto número de requisições HTTP paralelas aos 2 servidores e comparar o número de requisições atendidas por segundo. Também serão detalhados, em cada experimento, o percentual de uso do processador e a quantidade de memória gasta pelos softwares durante os testes, além de registrar quantas requisições não conseguiram ser atendidas com sucesso pelo servidor.

A máquina utilizada nos testes é uma máquina virtual com as seguintes configurações:

- Sistema Operacional: CentOS 7.
- Processador: Intel Core i5 2500K, 4 núcleos.
- Memória: 4 GB DDR3.
- Disco: SSD 240GB.

O software utilizado para realizar os testes é o weighttp. Ele é um pequeno software escrito pelos criadores do servidor web Lighttpd para testar a performance de servidores. O software consiste em um executável que envia diversas requisições simultâneas para simular o acesso em massa ao site. Os seguintes argumentos podem ser enviados ao executável

- -n <numero>. O número de requisições a serem enviadas. Obrigatório.
- -t <numero>. Número de threads a serem atualizadas. Por padrão é 1.
- -c <numero>. Número de clientes concorrentes. Por padrão é 1.
- -k. Para indicar se deve-se usar ou não conexões HTTP persistentes (keep-alive). N
- -6. Usar o protocolo IPv6 ao invés do IPv4.
- -h <string>. Para adicionar um cabeçalho extra à requisição, como por exemplo, um User-Agent ou um Cookie.

A escolha do weighttp, ao invés do mais conhecido ab (Apache HTTP Server Benchmarking Tool), se deve ao fato de que este último é capaz de utilizar somente 1 thread e utiliza um modelo de pooling de eventos ultrapassado, pois é um software antigo, criado em 2001, época em que não era tão comum processadores com vários núcleos. O weighttp é capaz de utilizar várias threads e é, portanto, mais apropriado para se testar máquinas com mais de um núcleo de processador, como a máquina utilizada nos testes deste trabalho. Além disso, o weighttp usa o epoll, um mecanismo notificação de eventos de I/O mais recente, introduzido no kernel 2.5.44 do Linux (KERRISK, 2015).

Foram realizados dois tipos de testes, um com requisições para conteúdo estático e outra para conteúdo dinâmico. Em cada teste foram feitas 100.000 requisições a uma página, variando-se o número de conexões simultâneas de 1 a 1.000.

O conteúdo estático consiste em baixar uma imagem de um wallpaper em resolução 1920x1200, com tamanho de 394 KB. O objetivo é simular vários clientes requisitando uma imagem em uma rede social.

O teste de conteúdo dinâmico é feito em um pequeno aplicativo, desenvolvido para os testes, que exibe a lista e o conteúdo de todas as obras de Shakespeare. O aplicativo é feito em PHP e as obras são consultadas em um banco de dados MariaDB, um fork do banco de dados MySQL. Os dados foram colhidos do projeto Open Source Shakespeare, disponíveis no repositório edent/Open-Source_Shakespeare no Github.

A aplicação oferece as seguintes urls que podem ser acessadas:

- /: Lista o título, ano e gênero de todas as obras de Shakespeare.
- /{nome-da-obra}: Lista todos os atos da obra selecionada.
- /{nome-da-obra}/{ato}: Lista todas as cenas do ato e obra selecionada.
- /{nome-da-obra}/{ato}/{cena}: Exibe o conteúdo da cena selecionada.

As urls retornam o conteúdo em formato HTML, mas pode-se também adicionar o sufixo .json ao final delas para se retornar o conteúdo em formato JSON, um popular formato de dados comumente usado para transmitir dados entre aplicações web e mobile.

O objetivo do aplicativo é simular algumas funcionalidades comuns em aplicativos web atuais, como conteúdo dinâmico com informações retiradas de um banco de dados e disponibilizadas via um web-service REST em formato JSON.

As versões do software utilizadas são as seguintes:

- Nginx: 1.9.3
- Apache: 2.4.6
- PHP: 5.4.16
- MariaDB: 10.0.20

Para a configuração do Nginx, utilizou-se os seguintes parâmetros.

- **worker_process**: 8. Número de processos do Nginx para atender as requisições. Foi escolhido o número de núcleos do CPU da máquina multiplicado por 2.

- **worker_connections**: 4096. Cada processo pode atender até 4096 conexões simultâneas.

Para a comunicação do Nginx com o PHP, utilizou-se o PHP em modo FastCGI através do PHP FastCGI Process Manager (PHP-FPM), configurado com os seguintes parâmetros

pm: static. Número fixo de processos.

pm.max_children: 100. Número máximo de 100 processos do PHP.

Para a configuração do Apache, testou-se o Apache utilizando o gerenciamento de requisições tanto em modo prefork como em modo worker e event. Em modo prefork utilizou-se os seguintes parâmetros:

- **MaxRequestWorkers**: 1000. Número máximo de conexões a serem processadas.
- **ServerLimit**: 1000. Número máximo de processos do Apache.
- **PHP** rodando via módulo mod_php.

Para a configuração do Apache em modo worker e event utilizou-se os seguintes parâmetros:

ServerLimit: 20. Número máximo de processos do Apache.

ThreadsPerChild: 50. Número máximo de threads a serem criadas por cada processo.

MaxClients: 1000. Número máximo total de threads a serem criadas.

PHP rodando via FastCGI com o PHP-FPM, assim como no Nginx.

Para a comunicação do Apache com o PHP, testou-se o mod_php, onde o interpretador PHP é embutido em cada processo do Apache, e testou-se o PHP em modo FastCGI, com as mesmas configurações utilizadas com o Nginx.

Resultados (Conteúdo Estático)

Nos testes de conteúdo estático, o Nginx apresentou um desempenho melhor em relação ao Apache (Gráfico 1), atingindo o valor máximo de 10.281 requisições por segundo (Tabela 1), contra 8.366 do Apache (Tabela 4). Verificou-se também que o Nginx sofreu pouca degradação de desempenho quando aumentado o número de conexões simultâneas, descendo apenas para 9.722 requisições por segundo nos testes com 1.000 conexões (Tabela 1). Já o Apache, mostrou uma grande degradação nessa situação, quando utilizados os antigos módulos mpm_prefork e mpm_worker, baixando, respectivamente, de 6.140 requisições por segundo para 2.201 (Tabela 2), e de 6.586 para 3.708 (Tabela 3), não conseguindo, inclusive, entregar

todas as respostas corretamente. Já com o módulo de gerenciamento de processos mais recente, o mpm_event, o Apache apresentou, assim como o Nginx, apenas uma leve perda de desempenho, baixando de 8.366 para 7.366 (Tabela 4), e entregando todas as respostas com sucesso. O uso da memória e da CPU foi bastante similar em todos os testes.

NGINX					
Requisições	**Concorrência**	**Requisições por Segundo**	**Uso Máximo de CPU**	**Uso de Memória (SO)**	**Respostas Erradas**
100.000	1	2.362 req/s	33,0%	115 MB	0
100.000	10	6.508 req/s	76,9%	118 MB	0
100.000	100	10.281 req/s	95,2%	123 MB	0
100.000	1.000	9.722 req/s	97,6%	150 MB	0

Tabela 1: Resultado dos testes de conteúdo estático com o Nginx.
Fonte: Elaboração própria (2015).

APACHE (mpm_prefork)					
Requisições	**Concorrência**	**Requisições por Segundo**	**Uso Máximo de CPU**	**Uso de Memória (SO)**	**Respostas Erradas**
100.000	1	1.773 req/s	28,4%	111 MB	0
100.000	10	5.804 req/s	83,3%	115 MB	0
100.000	100	6.140 req/s	80,0%	122 MB	0
100.000	1.000	2.201 req/s	81,2%	223 MB	259

Tabela 2: Resultado dos testes de conteúdo estático com o Apache + módulo mpm_prefork.
Fonte: Elaboração própria (2015).

APACHE (mpm_worker)					
Requisições	**Concorrência**	**Requisições por Segundo**	**Uso Máximo de CPU**	**Uso de Memória (SO)**	**Respostas Erradas**
100.000	1	1.596 req/s	26,9%	126 MB	0
100.000	10	5.879 req/s	87,0%	126 MB	0
100.000	100	6.586 req/s	84,0%	131 MB	0
100.000	1.000	3.708 req/s	88,3%	157 MB	60

Tabela 3: Resultado dos testes de conteúdo estático com o Apache + módulo mpm_worker.
Fonte: Elaboração própria (2015).

APACHE (mpm_event)					
Requisições	**Concorrência**	**Requisições por Segundo**	**Uso Máximo de CPU**	**Uso de Memória (SO)**	**Respostas Erradas**
100.000	1	1.726 req/s	31,7%	142 MB	0
100.000	10	6.702 req/s	93,4%	142 MB	0
100.000	100	8.366 req/s	97,0%	150 MB	0
100.000	1.000	7.366 req/s	99,1%	221 MB	0

Tabela 4: Resultado dos testes de conteúdo estático com o Apache + módulo mpm_event.
Fonte: Elaboração própria (2015).

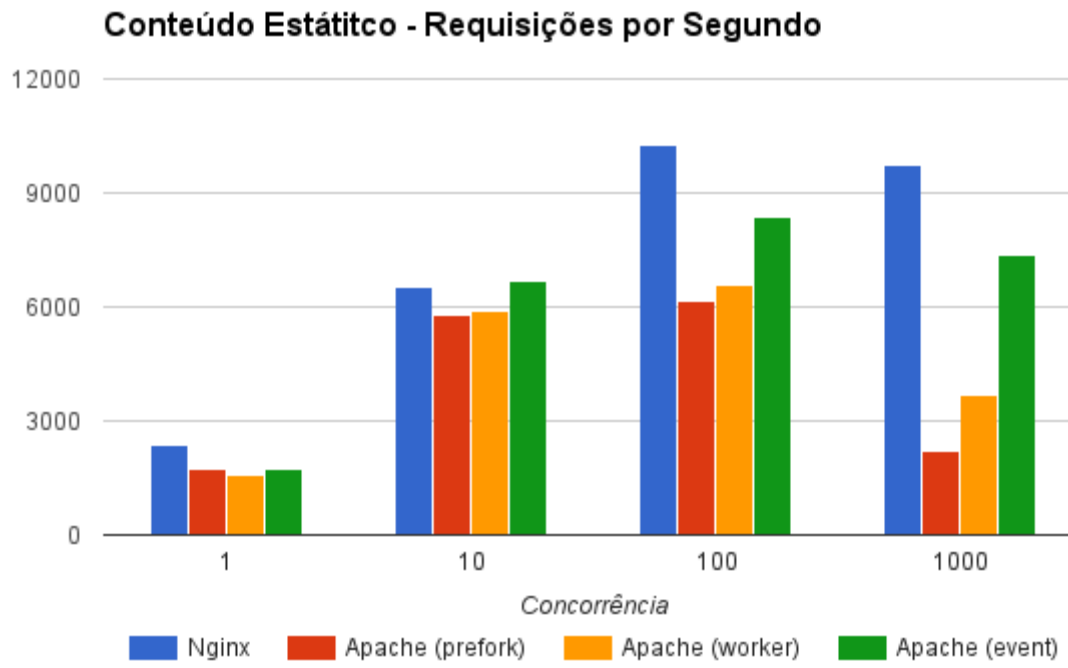

Gráfico 1 – Resultados dos testes de conteúdo estático.
Fonte: Elaboração Própria (2015).

Resultados (Conteúdo Dinâmico)

Nos testes de conteúdo dinâmico, a situação se repetiu (Gráfico 2). O Nginx e o Apache em modo mpm_event não sofreram uma degradação forte com 1.000 conexões simultâneas, com o Nginx descendo pouco, de 3.395 para 3.304 requisições por segundo (Tabela 5), e o Apache, mantendo-se praticamente estável, subindo de 2.849 para 2.866 (Tabela 7). Já com o módulo mpm_prefork, houve uma perda de desempenho alta, descendo de 3.629 para 1.865 requisições por segundo (Tabela 6), não conseguindo o Apache entregar todas as respostas com sucesso.

Um dado interessante a se notar, é que o Apache com o mod_php foi o que obteve o maior resultado nos testes, com 3.629 requisições por segundo, quando testado com 100 conexões simultâneas (Tabela 6). Isso se deve ao fato de que nesse módulo, o interpretador PHP está embutido dentro dos processos do Apache (LAYERSHIFT, 2015), o que evita o custo adicional de comunicação do servidor web com os processos PHP separados quando em modo FastCGI. Infelizmente, essa arquitetura não é escalonável e logo se percebe a piora de desempenho, quando aumentado o número de conexões.

NGINX + PHP (fastcgi)					
Requisições	**Concorrência**	**Requisições por Segundo**	**Uso Máximo de CPU**	**Uso de Memória (SO)**	**Respostas Erradas**
100.000	1	763 req/s	26,8%	313 MB	0
100.000	10	3.102 req/s	90,4%	315 MB	0
100.000	100	3.395 req/s	94,8%	315 MB	0
100.000	1.000	3.304 req/s	97,7%	347 MB	0

Tabela 5: Resultado dos testes de conteúdo dinâmico com o Nginx + PHP via FastCGI.
Fonte: Elaboração própria (2015).

APACHE + PHP (mpm_prefork + mod_php)					
Requisições	**Concorrência**	**Requisições por Segundo**	**Uso Máximo de CPU**	**Uso de Memória (SO)**	**Respostas Erradas**
100.000	1	818 req/s	23,4%	225 MB	0
100.000	10	3.538 req/s	91,4%	231 MB	0
100.000	100	3.629 req/s	92,6%	300 MB	0
100.000	1.000	1.865 req/s	95,4%	365 MB	402

Tabela 6: Resultado dos testes de conteúdo dinâmico com o Apache + mpm_prefork + mod_php.
Fonte: Elaboração própria (2015).

APACHE + PHP (mpm_event + fastcgi)					
Requisições	**Concorrência**	**Requisições por Segundo**	**Uso Máximo de CPU**	**Uso de Memória (SO)**	**Respostas Erradas**
100.000	1	646 req/s	28,4%	387 MB	0
100.000	10	2.781 req/s	98,1%	387 MB	0
100.000	100	2.849 req/s	99,5%	392 MB	0
100.000	1.000	2.866 req/s	99,8%	475 MB	0

Tabela 7: Resultado dos testes de conteúdo dinâmico com o Apache + PHP via módulo mpm_event + FastCGI.
Fonte: Elaboração própria (2015).

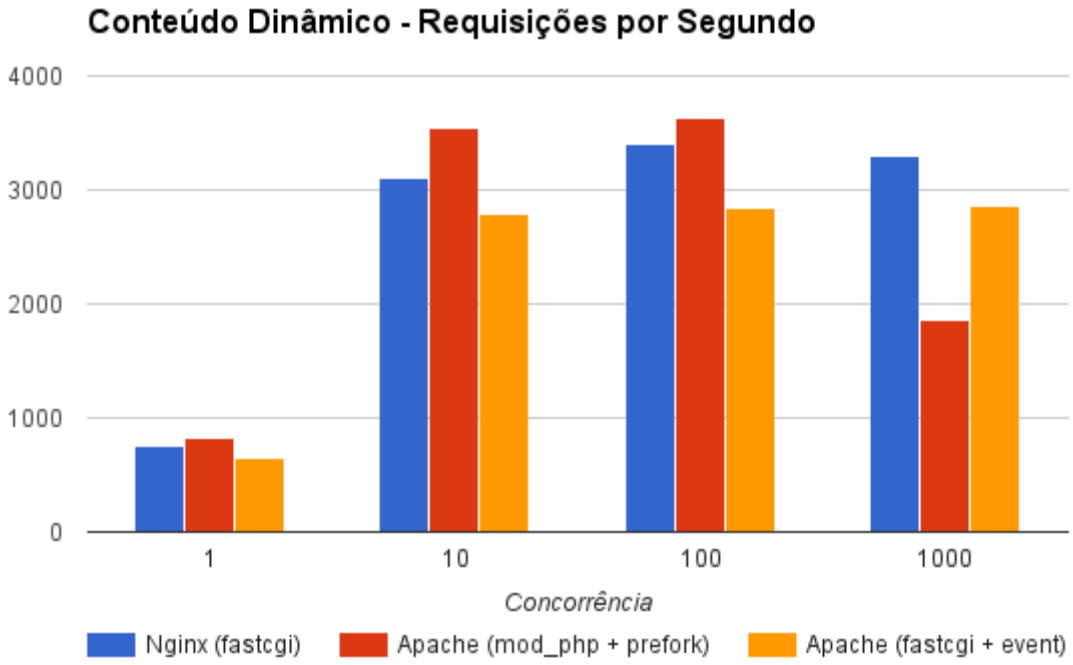

Gráfico 2: Resultados dos testes de conteúdo dinâmico.
Fonte: Elaboração Própria (2015).

7. Conclusão

Verificou-se, nos testes realizados, que o Nginx é a melhor opção de servidor web em relação a desempenho, liderando a maioria dos testes e mantendo seu resultado estável mesmo quando se aumenta o número de conexões simultâneas. O Apache, na sua última versão 2.4 e com o módulo mpm_event, também apresentou ótimos resultados, mantendo-se estável e respondendo a todas as requisições corretamente. A adoção de uma arquitetura assíncrona trouxe benefícios ao Apache, assim como ao Nginx, que já a usa por padrão.

Em relação às funcionalidades, os dois servidores são bem parecidos, com ambos suportando vários sistemas operacionais e provendo as principais funcionalidades como suporte a HTTPS, Virtual Hosting e suporte a CGI. Além disso, os dois softwares possuem uma arquitetura extensível, permitindo com que outros programadores adicionem funcionalidades extras aos servidores.

Como trabalhos futuros, seria interessante realizar os testes em máquinas mais poderosas, como aquelas encontradas nos centros de processamentos de dados de grandes empresas e órgãos públicos. Também seria interessante aumentar o número de conexões simultâneas para valores como 10.000 e até 100.000, e utilizar-se de uma aplicação de teste mais complexa, de modo a melhor emular os sistemas e a carga presentes nesses lugares.

Abstract

This work presents a comparison between the Apache HTTP Server, the 20 years old web server that current leads the market, and Nginx, the new web server that is being increasingly adopted by many system administrators in the world. The objective is to investigate if those web servers are able to fulfill the needs of modern information systems. We will make a comparison between the functionalities provided by each of these softwares, in addition to running some benchmarks in order to measure the performance of both servers when under heavy access load. At the end of this article, a conclusion is drawn indicating which software performed better.

Keywords: Web Servers. Nginx. Apache HTTP Server. Comparison. Performance. Benchmark.

8. Referências

AMY BROWN, G. W. **The Architecture of Open Source Applications, Volume II**. [S.l.]: lulu.com, 2012.

APACHE SOFTWARE FOUNDATION. About the Apache HTTP Server Project. **The Apache HTTP Server Project**, 2015. Disponivel em: <http://httpd.apache.org/ABOUT_APACHE.html>. Acesso em: 22 jul. 2015.

APACHE SOFTWARE FOUNDATION. Apache Projects Directory. **Projects by Name**, 2015. Disponivel em: <https://projects.apache.org/projects.html>. Acesso em: 22 jul. 2015.

APACHE SOFTWARE FOUNDATION. event. **The Apache HTTP Server**, 2015. Disponivel em: <http://httpd.apache.org/docs/2.4/mod/event.html>. Acesso em: 18 ago. 2015.

APACHE SOFTWARE FOUNDATION. prefork. **The Apache HTTP Server Project**, 2015. Disponivel em: <http://httpd.apache.org/docs/2.4/mod/prefork.html>. Acesso em: 18 ago. 2015.

APACHE SOFTWARE FOUNDATION. worker. **The Apache HTTP Server Project**, 2015. Disponivel em: <http://httpd.apache.org/docs/2.4/mod/worker.html>. Acesso em: 18 ago. 2015.

DALZIEL, H. **How to Attack and Defend Your Website**. 1ª. ed. [S.l.]: Syngress, 2015.

ERIC LARSON, B. S. **Administrating Web Servers, Security, & Maintenance Interactive Workbook**. [S.l.]: Prentice Hall, 1999.

INTERNET GUIDE. CERN httpd: the first web server software. **Internet Guide**, 2015. Disponivel em: <http://www.internet-guide.co.uk/CERNhttpd.html>. Acesso em: 15 out. 2015.

KEGEL, D. The C10K problem. **Dan Kegel**, 1999. Disponivel em: <http://www.kegel.com/c10k.html>. Acesso em: 22 jul. 2015.

KERRISK, M. epoll - I/O event notification facility. **Linux Programmer's Manual**, 2015. Disponivel em: <http://man7.org/linux/man-pages/man7/epoll.7.html>. Acesso em: 22 jul. 2015.

KRILL, P. Nginx boosts devops with custom JavaScript. **InfoWorld**, 2015. Disponivel em: <http://www.infoworld.com/article/2985457/javascript/nginx-custom-javascript-devops.html>. Acesso em: 15 out. 2015.

LAYERSHIFT. Which PHP mode? Apache vs CGI vs FastCGI. **Layershift**, 2015. Disponivel em: <http://blog.layershift.com/which-php-mode-apache-vs-cgi-vs-fastcgi/>. Acesso em: 19 ago. 2015.

NEDELCU, C. **Nginx HTTP Server, Second Edition**. [S.l.]: Packt Publishing Ltd., 2013.

NGINX INC. NGINX Inc. Launches NGINX Plus. **NGINX**, 2013. Disponivel em: <https://www.nginx.com/press/nginx-inc-launches-nginx-plus>. Acesso em: out. 2015.

NGINX INC. How NGINX Plans to Support HTTP/2. **NGINX**, 2015. Disponivel em: <https://www.nginx.com/blog/how-nginx-plans-to-support-http2/>. Acesso em: 15 out. 2015.

NGINX INC. NGINX Receives Fifth Consecutive Web Server of the Year Award from W3Techs. **NGINX**, 2015. Disponivel em: <https://www.nginx.com/blog/nginx-receives-fifth-consecutive-web-server-year-award-w3techs/>. Acesso em: 16 ago. 2015.

SYSOEV, I. History of Nginx. **Ruby Forum**, 2008. Disponivel em: <https://www.ruby-forum.com/topic/151853>. Acesso em: 15 out. 2015.

W3TECHS. Usage of web servers broken down by ranking. **W3Techs - World Wide Web Technology Surveys**, julho 2015. Disponivel em: <http://w3techs.com/technologies/cross/web_server/ranking>. Acesso em: 22 jul. 2015.

WEBOPEDIA. LAMP. **Webopedia:** Online Tech Dictionary for IT Professionals, 2015. Disponivel em: <http://www.webopedia.com/TERM/L/LAMP.html>. Acesso em: 15 out. 2015.