Um Estudo de Performance entre Servidor Web PHP e Node.js

Luana Martins dos Santos
Universidade Federal de Pernambuco
Centro de Informática
Recife, Pernambuco
Email: lms7@cin.ufpe.br

***Resumo*—Este artigo apresenta um estudo comparativo entre duas tecnologias para desenvolvimento *server-side* de serviços *web*. Entre elas, PHP, que é um framework bastante utilizado na indústria, enquanto que Node.js é mais recente. As tecnologias analisadas foram obtidas através de projetos do Github que realizam, entre outras funcionalidades, *download* de arquivos. Para analisar os servidores, foram feitas simulações de *download* de arquivos de tamanhos diferentes, considerando três variáveis, tempo de conexão, tempo de transferência e o total total da requisição. Obtidas essas métricas, foram feitas considerações sobre as variáveis necessárias para identificar qual dessas tecnologias, (...)**

1. Introdução

Com o avanço da internet, a utilização de *webservices* tornou-se uma forma muito interessante de prover serviços de maneira simples. Os usuários precisam apenas possuir um navegador ao qual podem acessar o serviço que precisam em qualquer computador, sem necessidade de instalá-lo. Portanto, o uso de tecnologias que auxiliem desenvolvedores a implementar tais serviços de maneira rápida e eficiente, torna-se um princípio bastante importante.

Avaliaremos duas tecnologias para desenvolvimento *web*, uma delas é o PHP, bastante conhecida na indústria, grandes empresas a utilizam ou utilizaram em algum momento, como Facebook, Google e Yahoo. Node.js, por outro lado, é um framework inovador, mais recente do que PHP, mas é considerado promissor por conta de suas características principais, visando escalabilidade e não-bloqueio de entrada e saída. Outro fator importante é que Node.js é escrito em JavaScript, uma linguagem de programação bem estabelecida em desenvolvimento *web*. Este estudo pretende comparar as tecnologias PHP e Node.js em aspectos de concorrência, tempo de resposta (para o envio de requisições) e principalmente funcionalidades de entrada e saída, como download de arquivos.

O foco deste estudo é verificar as diferenças de performance entre servidores Node.js e PHP para entrada e saída (a avaliação é de envio de arquivos). Identificar em qual aspecto os servidores podem ser mais rápidos, podendo auxiliar na escolha de uma tecnologia para desenvolvimento *web*. Utilizamos basicamente dois cenários de teste, considerando carga e concorrência.

O restante deste artigo está organizado da seguinte maneira, a seção ...(Explicar como o artigo está dividido considerando as seções)

2. Trabalhos relacionados

Todo Comparar esses dois projetos [1] [2]

3. Apresentação das tecnologias

Para analisar tecnologias que auxiliem o desenvolvimento *web*,é preciso primeiramente realizar algumas considerações sobre o que ou quais conceitos devem ser avaliados para a escolha de uma dada tecnologia em detrimento de outra. É importante frisar que as tecnologias avaliadas nesse estudo já são conhecidas pela comunidade de desenvolvimento. Cada uma delas possui vantagens e desvantagens, considerando escalabilidade, performance, ou mesmo concorrência. (...)

3.1. Node.js

Node.js é um *framework* criado para desenvolvimento do lado servidor, utilizando uma das mais usadas linguagens de programação, *JavaScript*. Esse *framework* utiliza de um modelo não bloqueante para requisições de entrada e saída (IO), altamente direcionada a eventos, tornando uma tecnologia leve e eficiente. Também utiliza o *npm* como gerenciador de bibliotecas *open-source online*, assim como o é Maven para Java. Foi construído utilizando a engine JavaScript V8 do Chrome da Google [4].

3.2. PHP

Como Node.js, PHP é também uma linguagem de script, voltada para desenvolvimento *web* [5]. Proposta para desenvolvimento script de páginas dinâmicas, incluindo CLI (interface de linha de comando) e GUI (graphical user interface) [1]. (verificar algumas características do Apache) (...)

4. Ambiente dos experimentos

Para simulação dos experimentos, considera-se dois equipamentos onde um deles possuirá os *webservers* Node.js e PHP executando, enquanto o segundo equipamento irá realizar as medições através de uma biblioteca (cURL). As chamadas dessa ferramenta foi desenvolvida em Python, utilizando a biblioteca pycurl. Node.js é um servidor *standalone*, não sendo necessário o uso de ambientes auxiliares para permitir que o servidor estivesse online. Não é o caso, porém, do PHP, onde foi utilizado o Apache Tomcat, versão 8, para permitir que os experimentos fossem executados utilizando essa tecnologia.

4.1. Hardware utilizado

As configurações das máquinas foram as seguintes: a primeira, utilizada para obter as medições foi uma Intel(R) Core(TM), i5-3210M x64, 2.5 GHz, 6GB (RAM), 800 GB de disco usando sistema operacional Windows 10. Enquanto a máquina, com os servidores disponíveis, AMD E1-500 APU com Radeon HD Graphics, 1.48 GHz, 2GB (RAM), 500 GB de disco. Os testes foram realizados utilizando a mesma rede e nenhuma outra aplicação estava rodando enquanto os servidores estavam executando. Inclusive, os servidores não executaram ao mesmo tempo durante os testes.

4.2. Configuração dos servidores

A versão utilizada no Node.js foi a 4.4.5, enquanto PHP foi a versão 7.0.6. É importante frisar que foi necessário também o uso do Apache Tomcat (foi utilizada a versão 8 para que possa tornar o servidor PHP disponível).

4.3. Ferramentas de medição

Para realizar as medições, foi utilizada a ferramenta cURL [3], que transfere ou recebe dados de servidores, considerando uma quantidade de protocolos pré-estabelecidos. Esta foi escrita em C, porém nesse estudo utilizou-se a biblioteca ***pycurl***, que é um *wrapper* para o *libcurl* (biblioteca do cURL) desenvolvida em C. O cURL possui opções que são adicionadas no envio do comando que o usuário deseja realizar. É possível verificar a lista de opções disponíveis na página do cURL [3].

Explicar as tecnologias curl e python(apresentar o código feito) utilizado nas medições (...)

5. Metodologia dos experimentos

As medições foram divididas em duas partes considerando carga (tamanho do arquivo a ser baixado) e concorrência. Nesse caso, concorrência será considerada um conceito onde uma quantidade de usuários realiza a mesma requisição ao mesmo tempo. A biblioteca do cURL possui em uma das suas opções, o *-w* que permite escrever alguns parâmetros obtidos das requisições para arquivo. São considerados 3 parâmetros de retorno quando um requisição é feita através do cURL [3].

1) Tempo de conexão: É definido pela constante CONNECTION_TIME, que o curl retorna o tempo em segundos (s).

2) Velocidade de transferência: É definido pela constante SPEED_DOWNLOAD, que o curl retorna o tempo em bytes por segundo (B/s).

3) Tempo total: É definido pela constante TOTAL_TIME, que o curl retorna o tempo total da requisição em milissegundos (ms).

5.1. Primeiro experimento

Inicialmente, criamos os recursos a serem requisitados dos servidores, estabelecendo que dois arquivos seriam enviados através das requisições, um arquivo de 100 páginas e outro de 1000 páginas. O conteúdo de cada arquivo foi gerado a partir de um *website* de *lorum ipsum* [6] e replicado até atingir o total de páginas estabelecido.

Para o experimento 1, as requisições foram feitas de forma sequencial, primeiro para o PHP e depois para o Node.js, com um total de 500 requisições para cada um dos servidores. Essas informações foram coletadas pelo programa em Python desenvolvido (...)

5.2. Segundo experimento

Os arquivos utilizados nos testes anteriores foram também utilizados como *resources* para esse segundo teste, porém as requisições foram concorrentes. Para isso, foi necessário o uso do objeto CurlMulti, pois o uso de threads em python não iria simular um comportamento concorrente de forma efetiva, fazendo com que a biblioteca executasse as requisições de forma sequencial em C. A quantidade de requisições foi a mesma do primeiro cenário de teste.

6. Resultado dos experimentos

Após a definição dos testes e cenários, e realizadas as medições, foram obtidos 12 arquivos, considerando cada uma das opções do cURL consideradas nesse estudo (tempo total, tempo de conexão e tempo de transferência), variando a *resource* (um arquivo de 100 ou 1000 páginas), para cada um dos servidores testados (Node.js e PHP). Com a posse de dados, é necessário explorar os dados obtidos para identificar como estes estão distribuídos e assim compará-los visualmente.

6.1. Análise exploratória dos dados

Foram criados *boxplots* para cada uma das amostras a título de comparação de médias e variâncias entre *webservers*. Histogramas foram criados de forma isolada para cada

conjunto de dados, diferentemente da criação dos *boxplots*. Esses gráficos foram gerados com o auxílio da ferramenta R, no R Studio. Houve um pré-processamento dos dados coletados no que tange à variável SPEED_DOWNLOAD, como os valores são obtidos em B/s (bytes por segundo), para facilitar a visualização e análise dos boxplots, os valores foram atualizados para KB/s (quilobytes por segundo). Para facilitar o entendimento, vamos considerar nas próximas seções que as requisições de arquivos com 100 páginas pertencem ao cenário de teste 1, enquanto o envio de arquivos com 1000 páginas pertence ao cenário de teste 2.

6.1.1. Tempo de conexão. Para a primeira variável, as medianas para envio de arquivos (com 100 e de 1000 páginas) dos servidores Node.js e PHP, apresentaram-se de forma similar. Apesar de existir uma quantidade alta de *outliers*, tanto para o envio de um arquivo de 100 páginas quanto o arquivo de 1000 páginas. As medianas podem ser comparadas visualmente utilizando boxplots, como são apresentados através das figuras 1, 2. Na figura 3 é possível visualizar os tempos de conexão para o envio de 100 páginas, com um aumento sobre a mediana e quartis, para melhor identificar as diferenças existentes. A diferença não é muito perceptível na primeira imagem (figura 1) por conta da existência de um *outlier*, um valor para a variável CONNECT_TIME de três segundos.

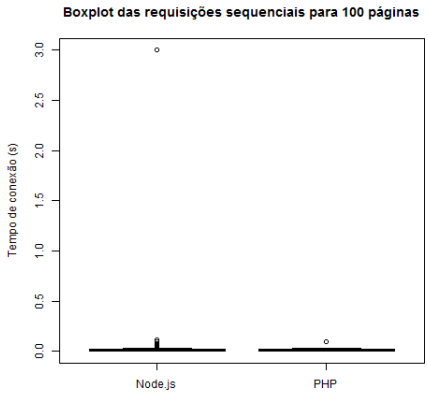

Figura 1. Boxplots de Node.js e PHP considerando tempo de conexão

A comparação anterior foi relativa às requisições quando realizadas de forma sequencial, porém ao analisar as requisições feitas de forma concorrente, o comportamento verificado foi diferente. O servidor Node.js possui uma mediana maior que o servidor PHP, considerando o envio de 100 páginas do arquivo de teste, para o tempo de conexão. O envio de um arquivo de 1000 páginas, porém, mostrou uma mudança nesse quadro, tal que o servidor Node.js obteve uma mediana de tempo de conexão mais abaixo da mediana do servidor PHP. O contraste identificado é visualizado nas figuras 4 e 5.

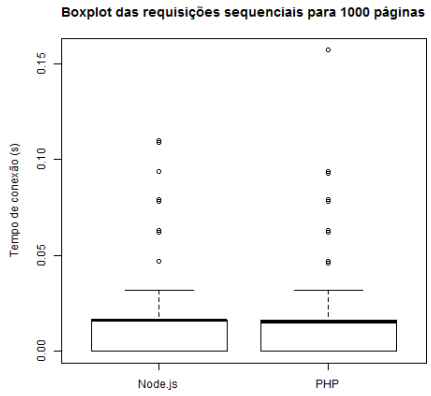

Figura 2. Boxplots de Node.js e PHP considerando tempo de conexão

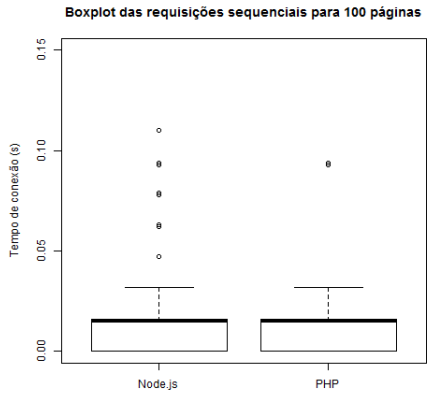

Figura 3. Boxplots de Node.js e PHP considerando tempo de conexão, com amplificação das médias

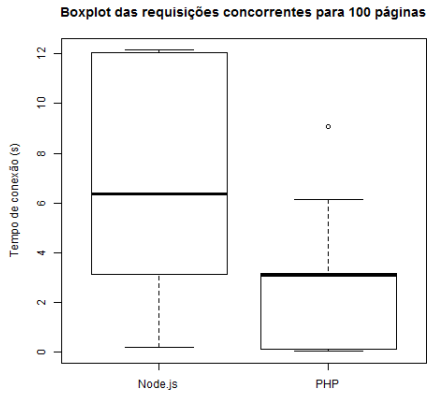

Figura 4. Boxplots de Node.js e PHP considerando tempo de conexão

6.1.2. Velocidade de transferência. Para a variável aleatória SPEED_DOWNLOAD, consideraremos que quanto mais alta for o seu valor, de maior performance

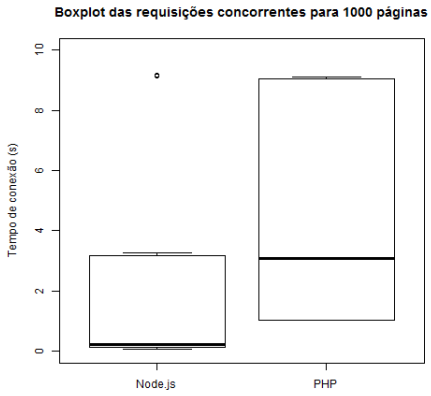

Figura 5. Boxplots de Node.js e PHP considerando tempo de conexão

o servidor poderá ser considerado. O que realmente faz sentido, pois indica que os usuários que requisitaram os arquivos podem ter o tempo de espera pelo arquivo reduzido.

Para os valores encontrados, no primeiro cenário de teste, as medianas apresentaram-se muito próximas para os ambos os servidores. Os boxplots para este cenário, como ser visualizado na figura 6, onde o servidor de tecnologia PHP possui mediana de 650 KB/s e enquanto Node.js aproximadamente 670 KB/s. Pode-se visualizar uma quantidade de *outliers* muito grande para ambos os servidores também. Porém os dados estão mais ”espalhados”quando o servidor em Node.js é utilizado. Vale ressaltar porém que a mediana das requisições em Node.js possui um valor aproximadamente o mesmo do maior valor encontrado para PHP, neste mesmo cenário de teste.

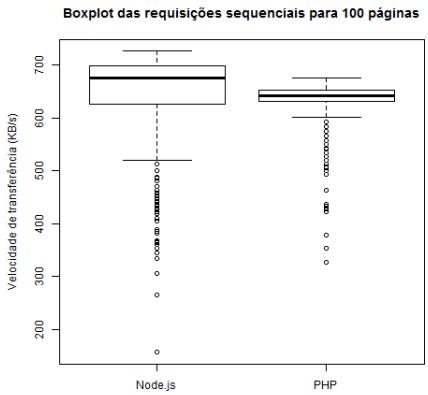

Figura 6. Boxplots de Node.js e PHP considerando velocidade de transferência

(Analisar o boxplot para 1000 páginas - sequential)

Um cenário um pouco diferente é percebido quando são feitas requisições para arquivos de 1000 páginas.

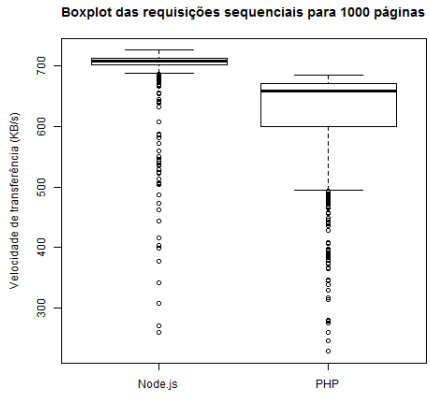

Figura 7. Boxplots de Node.js e PHP considerando velocidade de transferência

Vamos considerar o cenário de teste 1, inicialmente, e podemos visualizar na figura 8, que a mediana de ambos os servidores é próxima, com alguns *outliers* para Node.js e um quartil superior maior para o servidor PHP, indicando que temos casos que demoraram mais para realizar o *download*. Um ponto interesse acontece quando aumentamos a figura 6, para comparar as diferenças entre as medianas de uma forma mais apropriada. Essa diferença é vista na figura 9, nota-se uma discrepência muito maior do que vista inicialmente. Sendo a mediana de Node.js um pouco maior que 0.5 KB/s e PHP, um pouco acima 2 KB/s, sendo portanto um pouco mais rápido.

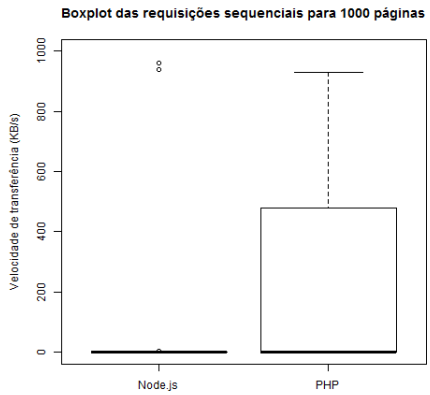

Figura 8. Boxplots de Node.js e PHP considerando velocidade de transferência

Para o segundo cenário, (...)

6.1.3. Tempo total. Todo

6.1.4. Comparação das médias das variáveis aleatórias. Ao comparar apenas as médias para as categorias de envio de arquivos de 100 páginas considerando sem e com con-

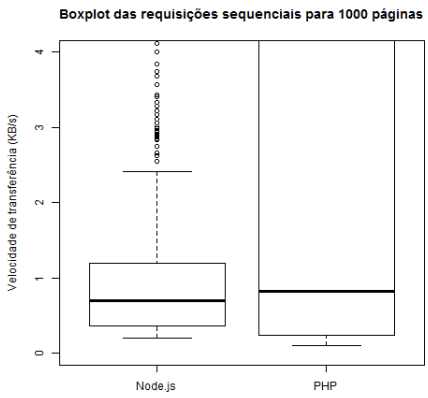

Figura 9. Boxplots de Node.js e PHP considerando velocidade de transferência

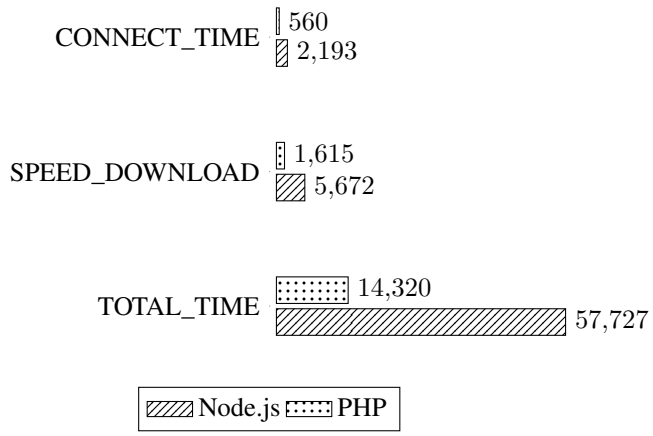

Figura 10. Gráficos de médias das requisições sequenciais

corrência.

Teste para referencia 10

6.2. Testes de hipótese

Podemos verificar pelos boxplots gerados que nenhuma das amostras coletadas possui distrubuição normal. Também foram construídos histogramas sobre os dados das amostras mas estes foram omitidos desse artigo. Notamos que a distribuição dos dados não é normal para nenhuma das amostras coletadas. Assim, iremos utilizar o teste de Wilcoxon Signed-Rank, considerando além da não-normalidade dos dados, a natureza pareada em que os dados foram coletados.

É razoável considerar que o servidor em Node.js tenha uma melhor performance em comparação a um servidor desenvolvido em PHP, principalmente pela natureza do servidor ser *event-driven* para casos de entrada e saída. Para testar essa consideração, vamos utilizar as variáveis descritas anteriormente (CONNECT_TIME, SPEED_DOWNLOAD, TOTAL_TIME)

É necessário identificar primeiramente a distribuição que se adeque aos conjuntos de dados. - Teste de aderência dos dados - Teste de homogeneidade das variâncias - Teste de hipóteses

Hipótese 1 (H1): *This is my first hypothesis.*

Hipótese 2 (H2): *This is my second hypothesis.*

$$H_0 : \mu_X = \mu_Y$$
$$H_1 : \mu_X \neq \mu_Y$$

7. Conclusão

Esse artigo apresenta um estudo comparativo entre as tecnologias Node.js e PHP para desenvolvimento de servidores *web*.
Em resumo, (...)

Referências

[1] Kai Lei, Yining Ma, Zhi Tan, *Performance Comparison and Evaluation of Web Development Technologies in PHP, Python and Node.js*. IEEE 17th International Conference on Computational Science and Engineering, 2014.

[2] Ioannis K. Chaniotis, Kyriakos-Ioannis D. Kyriakou, Nikolaos D. Tselikas, *Is Node.js a viable option for building modern web applications? A performance evaluation study*. Springer-Verlag Wien, 2014.

[3] https://curl.haxx.se/docs/manpage.html

[4] Node.js, https://nodejs.org/en/

[5] PHP, http://php.net/

[6] http://www.lipsum.com/