Cluster com Node.

Eduardo Bianchi[1], Emerson Silva[2]

1. Introdução

Node se trata de um ambiente que permite a execução de códigos Javascript do lado do servidor, além de ser altamente escalável, é de simples utilização. Este documento visa demonstrar passo a passo, a configuração de um cluster utilizando Node, bem como analisar o seu funcionamento.

2. Ferramentas utilizadas

Nesta seção serão citadas as ferramentas utilizadas para a configuração e monitoramento do ambiente.

2.1 Cluster

Cluster é um módulo nativo do Node que permite que processos filhos sejam criados para a realização de tarefas em paralelo.

2.2 Load Balancer NGINX

O NGINX é responsável por realizar o balanceamento de carga e distribuir as chamadas de serviços entre os processos e CPU's que são utilizadas pelo sistema. Existem diferentes configurações que podem ser definidas para o balanceamento.

2.3 Gerenciador de servidores PM2

O PM2 é um módulo que permite o gerenciamento de servidores node, ou seja, permite iniciar, parar e monitorar servidores node. Ele será utilizado para iniciar em segundo plano alguns servidores além de servir como monitor de consumo de recursos computacionais e monitor de status.

2.4 Monitor de Processos

System information é um pacote que permite a coleta de dados do sistema como por exemplo uso de CPU, memória, bateria, discos/filesystem além de processos e serviços. Essa biblioteca possui mais de trinta e cinco funções para a obtenção detalhada de dados do sistema operacional e de hardware.

2.5 Artillery

O teste de stress será realizado com o auxílio da ferramenta Artillery, que permite a realização de diversas chamadas simultâneas de um serviço durante um determinado período de tempo.

Após a realização das chamadas, os dados de resposta do servidor serão

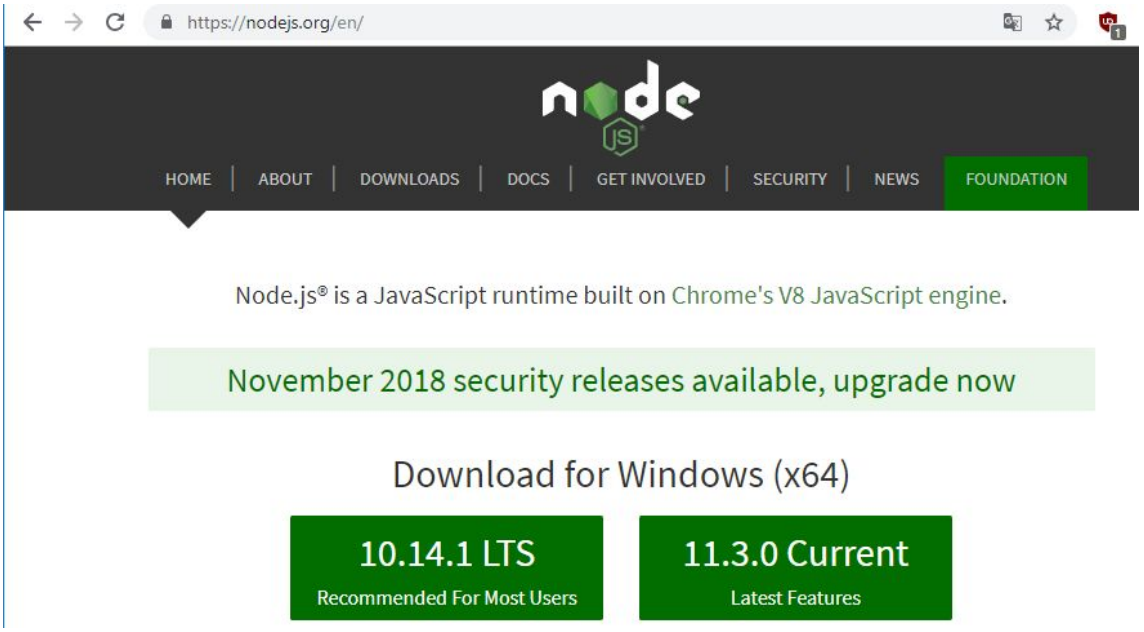
exibidos. Este processo é importante para testar e analisar o desempenho de chamadas de determinados serviços.

3. Configuração do Ambiente

Nesta seção serão demonstrado os passos de instalação do ambiente. Durante este processo estaremos utilizando o editor de texto Visual Code.

3.1 Instalação NodeJS

A NodeJS pode ser transferido a partir de sua pagina oficial https://nodejs.org. Basta acessar a página inicial e realizar o download da versão desejada. Durante a execução deste documento utilizamos a versão 10.14.1 LTS.



Página de download NodeJS

Após o download do NodeJS, realizaremos a sua instalação. Para fazer isso basta abrir o executável, escolher o diretório de instalação e aguardar a mesma ser finalizada.

3.2 Criação do Projeto e Instalação de Ferramentas.

Após a instalação do Node JS, será criada uma pasta para o projeto. Após a criação desta pasta, deve-se criar um arquivo dentro dela chamado “server.js”, que irá conter os códigos fonte do servidor.

Criado o arquivo, será feita a instalação das ferramentas citadas na seção dois deste documento. Para isso deve-se abrir a pasta do projeto no console do sistema e digitar os seguintes comandos descritos nas imagens, um de cada vez.

Primeiro será fornecido o comando **npm init** e após a execução do mesmo, deve-se informar os dados básicos do projeto, como descrição, nome e versão.

Para a instalação dos módulos, executaremos os seguintes comandos, um de cada vez: **npm i artillery**, **npm i express** e **npm i pm2**. Com todos os módulos instalados, deve-se abrir o arquivo criado previamente "server.js" e copiar o código fonte do servidor, que está anexado junto com este documento.

O *link* de *download* do NGINX pode ser encontrado na página de *downloads* do site oficial do NGINX (https://nginx.org/en/download.html). Será realizado download de um arquivo zipado (.zip), todo seu conteúdo deve ser extraído no caminho C:\PROGRA~1\. Após extrair os arquivos, deve-se editar o arquivo **nginx.conf** adicionando as configurações destacadas na imagem abaixo.

Nesse exemplo será utilizado a configuração padrão, mais opções de configurações podem ser encontradas na documentação do NGINX (http://nginx.org/en/docs/beginners_guide.html).

```
28      #tcp_nopush     on;
29
30      #keepalive_timeout  0;
31      keepalive_timeout  65;
32
33      #gzip  on;
34
35      upstream my_http_servers {
36          least_conn;
37          server 127.0.0.1:8001;      # httpServer1 listens to port 444
38          server 127.0.0.1:8002;      # httpServer2 listens to port 445
39          server 127.0.0.1:8003;      # httpServer3 listens to port 446
40          server 127.0.0.1:8004;      # httpServer4 listens to port 447
41      }
42
43      server {
44          listen       80;
45          server_name  localhost;
46
47          #charset koi8-r;
48
49          #access_log  logs/host.access.log  main;
50          location / {
51              proxy_set_header   X-Real-IP $remote_addr;
52              proxy_set_header   Host      $http_host;
53              proxy_pass         http://my_http_servers;
54          }
55
56          #error_page  404              /404.html;
```

5. Iniciar aplicação

Para iniciar o servidor deve-se utilizar o **prompt de comando**, abrir a pasta onde o projeto foi criado e executar os comandos **pm2 delete all** e **pm2 start server.js**, esses comandos devem parar qualquer servidor node em execução e iniciar a aplicação simples. (Verificar capítulo **6. Stress Test** para monitorar e testar o servidor).

É possível observar que o servidor demora em média aproximadamente **4200 ms** para responder a 99% das requisições e com o menor tempo de **109 ms** e máximo de **4700 ms** com a configuração de teste demonstrada no capítulo **6. Stress Test.**

Para iniciar a aplicação utilizando clusters executar os comandos **pm2 delete all** e **pm2 start clusters.js**, esses comandos devem parar qualquer servidor node em execução e iniciar a aplicação de clusters.

É possível observar que o servidor demora em média aproximadamente **1180 ms** para responder a 99% das requisições e com o menor tempo de **117 ms** e máximo de **5773 ms** com a configuração de teste demonstrada no capítulo **6. Stress Test.**

Para iniciar a aplicação utilizando 4 servidores e o NGINX como LoadBalancer é necessário executar os comandos **pm2 delete all, pm2 start nginx_01.js, pm2 start nginx_02.js, pm2 start nginx_03.js** e **pm2 start nginx_04.js** esses comandos devem parar qualquer servidor node em execução e iniciar 4 novos servidores, esses servidores ainda atuam individualmente então é necessário iniciar o NGINX, para isso basta executar o arquivo **nginx.exe**, anteriormente descompactado durante a instalação do NGINX, ele pode ser encontrado na pasta **C:\PROGRA~1\nginx-1.14.1**.

É possível observar que o servidor demora em média aproximadamente **1200 ms** para responder a 99% das requisições e com o menor tempo de **120 ms** e máximo de **5800 ms** com a configuração de teste demonstrada no capítulo **6. Stress Test.**

6. Stress Test

Abrir o **prompt de comando** e digitar o comando **artillery quick --count 40 -n 40 http://localhost:8000/**.

O módulo artillery permite coletar alguns dados interessantes sobre o tempo de resposta da aplicação, dentre eles o número de requisições atendidas, tempo mínimo de resposta, tempo máximo de resposta e tempo médio de resposta.

7. Conclusões

Após a execução do ambiente, pode-se perceber que utilizando um servidor sem o módulo de clusters, o sistema fica lento e com baixa disponibilidade, uma vez que qualquer requisição que possa ocupar mais recursos computacionais vai causar travamentos, visto que apenas uma thread está tratando do processamento dessas requisições.

Ao utilizar clusters, a taxa de resposta fica mais rápida pois mais threads são criadas para lidar com o número de requisições, os tempos de resposta mínimo e máximo foram elevados um pouco, porém o tempo médio de resposta desceu significativamente.

O NGINX não demonstrou bons resultados em relação a aplicação utilizando o módulo de clusters, isso porque os 4 Núcleos do processador estavam ocupados da mesma forma como na aplicação com clusters, porém o NGINX permite realizar o balanceamento de cargas, não apenas entre servidores rodando na mesma máquina, mas

também com servidores rodando em máquinas diferentes, além de prover outras opções de balanceamento, tais como peso de cada nó e algoritmo de balanceamento.