21212 Academy

Medindo desempenho do Wordpress e outras aplicações PHP

Intro

Uma das plataformas que costumo usar para iniciar projetos é o Wordpress. Quem me conhece sabe sobre minha "paixão" por PHP (#sqn). Então por que uso Wordpress em projetos? Pela facilidade de se encontrar profissionais (tanto de Design como de Desenvolvimento) que já tenham mexido com ele.

Só que depois de um tempo adicionando plugins, tunando seu tema, engordando seu funcitons.php seu site começa a ficar lento. Até mesmo a interface administrativa começa a ficar insuportavelmente lenta e você não faz idéia do que pode estar causando essas lerdadas.

A proposta deste artigo é abrir esta caixa opaca e mostrar como você pode investigar os gargalos do seu site (tanto Wordpress como qualquer aplicação PHP) tanto em desenvolvimento quanto em produção.

O que vamos fazer

Para realizar esta investigação temos que efetuar 2 passos:
habilitar a instrumentação
analisar os dados coletados

1. Habilitando a instrumentação

Habilitando instrumentação do PHP

Para poder gravar os dados sobre as instruções que estão sendo executadas no seu código temos que instalar a extensão Xdebug no nosso ambiente.

No Ubuntu Linux instala-se possivelmente da seguinte forma:

```
sudo apt-get install php5-xdebug
```

No OSX com Homebrew assim, por exemplo:

```
brew install php55-xdebug
```

Para permitir o uso do profiler (o cara que faz as medições de desempenho na sua aplicação) é necessário adicionar estas linhas no arquivo de configuração do xdebug (Linux: /etc/php5/apache2/conf.d/20-xdebug.ini, OSX: /usr/local/etc/php/5.5/conf.d/ext-xdebug.ini):

```
; profiler
xdebug.profiler_enable_trigger = 1
xdebug.profiler_output_dir = "/tmp"
```

Depois disso reinicie o apache ou o php-fpm, dependendo do seu caso.

Realizando sua primeira medição

Agora que o profiler está habilitado, escolha uma página que esteja apresentando uma lentidão bem perceptível e adicione o parâmetro ?XDEBUG_PROFILE=1:

http://example.com/?XDEBUG_PROFILE=1

Assim que a página aparecer, serão criados na pasta /tmp do seu servidor um ou mais arquivos cachegrind:

```
$  ls -l /tmp/cache*
-rw-r--r--   1   _www   wheel   3511411   Jun   10   13:46
/tmp/cachegrind.out.43324
-rw-r--r--   1   _www   wheel   3470572   Jun   10   13:46
/tmp/cachegrind.out.43337
```

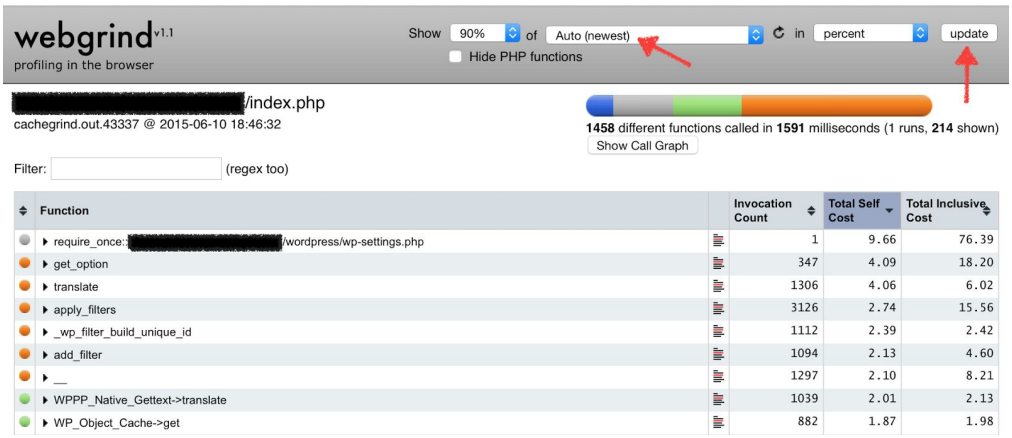
E é isso! Agora utilizaremos o projeto Webgrind para ler o conteúdo destes arquivos!

2. Analisando os dados coletados

Agora que sua sessão foi medida e os dados estão guardados na pasta /tmp vamos interpretá-los através de uma ferramenta de uso super simples, o Webgrind. Basta fazer o download dele e colocar em alguma basta qualquer que seja acessível no seu servidor. Por exemplo sua pasta public_html:

http://example.com/~seu_login/webgrind/

Você vai ver uma tela como essa:



Para começar basta clicar em update que ele vai pegar o arquivo cachegrind mais recente da pasta /tmp.

A primeira coluna explica o tipo de chamada que foi feita: cinza = include/require, laranja = chamada de função, azul = função nativa, verde = chamada de método
A segunda coluna Function contém o nome da função/método/include que foi chamado
A terceira coluna possui um link que leva para a linha do arquivo correspondente (super útil!)
A quarta coluna Invocation Count contém o número de vezes que esta chamada foi feita durante o request analisado
A quinta coluna Total Self Cost contém o tempo necessário para executar esta função/método/include desconsiderando as chamadas que ele realizou
A sexta coluna Total Inclusive Cost é igual a anterior porém soma também o tempo de execução das chamadas que esta função/método/include realizou

Você pode usar estas informações tanto pra 'escovar bit' e se aprofundar no porque das lerdezas da sua aplicação quanto para ter uma idéia de alto nível sobre quais plugins são responsáveis pelos maiores tempos de execução.

Na minha última investigação por exemplo, a função que estava causando os maiores tempos de execução era a gettext.

Após uma busca rápida por arquivos ".po" e ".mo" percebi que havia um arquivo gigante herdado de um teste que fizemos com o plugin woocommerce (que nem estávamos usando mais...). Removi estes arquivos e o gargalo desapareceu ;)

Ah, e depois de terminarem seus testes, comentem as linhas que você adicionou no arquivo .ini do xdebug!

Se tiverem idéias para outros posts, aceito sugestões nos comentários abaixo.

abs, Cyber