Web 開發的重要概念辨析：CGI、WSGI、uWSGI、ASGI.....

DevOps在路上 濤哥聊Python 2021-12-10 13:00

作者：DevOps在路上

來源：https://www.cnblogs.com/FLY_DREAM/p/15635690.html

在學習Python Web 開發時候，可能會遇到諸如uwsgi、wsgi 等名詞，下面通過梳理總結，探究它們之間的關係。

CGI

CGI（Common Gateway Interface）通用網關接口，是一個協議，是外部應用程序（CGI 程序）與Web 服務器之間的接口標準，該協議定義了Web 服務器調用外部應用程序的時候需要輸入的參數，和給Web 服務器的返回結果。

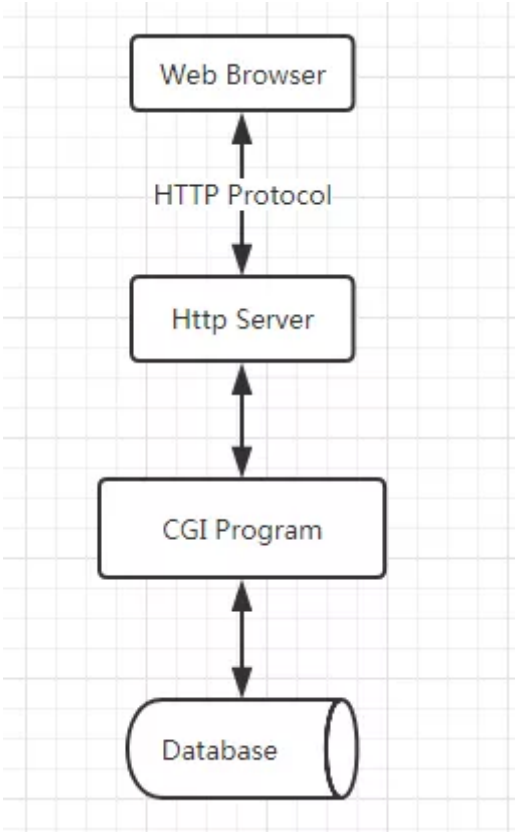
通俗來說，它規定一個程序該如何與Web 服務器程序之間通信，從而可以讓這個程序跑在Web 服務器上。

起源

最早的Web 服務器簡單地響應瀏覽器發來的HTTP 請求，並將存儲在服務器上的HTML 文件返回給瀏覽器，也就是靜態HTML。這個場景下的服務器一般被稱為HTTP 服務器，常見的有Apache 的httpd 和Nginx。

事物總是不斷發展，網站也越來越複雜，所以出現動態技術。但是服務器並不能直接運行php、asp這樣的文件，自己不能做，外包給別人吧，但是要與第三做個約定，我給你什麼，然後你給我什麼，就是握把請求參數發送給你，然後我接收你的處理結果給客戶端。

那這個約定就是Common Gateway Interface，簡稱CGI。這個協議可以用VB、C、PHP、Python來實現。CGI只是接口協議，根本不是什麼語言。

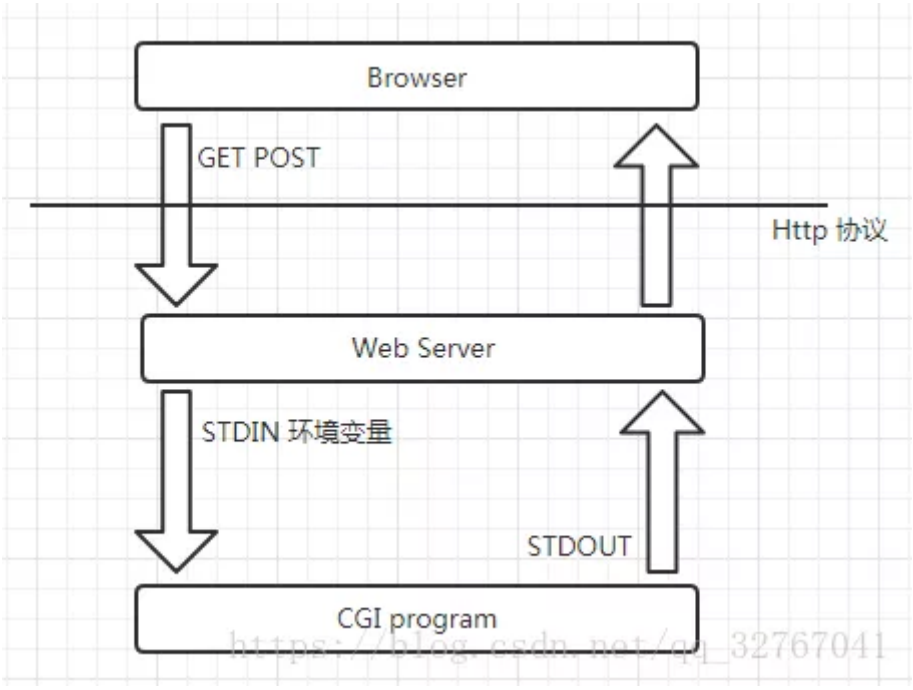


引入CGI 以便客戶端請求能夠觸發Web 服務器運行另一個外部程序，客戶端所輸入的數據也會傳給這個外部程序，該程序運行結束後會將生成的HTML 和其他數據通過Web 服務器再返回給客戶端（即動態請求，比如基於PHP、Python、Java 實現的應用）。利用CGI 可以針對用戶請求，動態返回給客戶端各種各樣動態變化的信息。

工作原理

Web 服務器與CGI 程序的交互

Web 服務器將根據CGI 程序的類型決定數據向CGI 程序的傳送方式，一般是通過標準輸入/輸出流和環境變量來與CGI 程序間傳遞數據。如下圖所示：



CGI 程序通過標準輸入（STDIN）和標準輸出（STDOUT）來進行輸入輸出。此外CGI 程序還通過環境變量來得到輸入，操作系統提供了許多環境變量，它們定義了程序的執行環境，應用程序可以存取它們。Web 服務器和CGI 接口又另外設置了一些環境變量，用來向CGI 程序傳遞一些重要的參數。

常用CGI 環境變量：

變量名	描述
CONTENT_TYPE	這個環境變量的值指示所傳遞來的信息的MIME類型。目前，環境變量CONTENT_TYPE一般都是：application/x-www-form-urlencoded,他表示數據來自於HTML表單。
CONTENT_LENGTH	如果服務器與CGI程序信息的傳遞方式是POST，這個環境變量即使從標準輸入STDIN中可以讀到的有效數據的字節數。這個環境變量在讀取所輸入的數據時必須使用。
HTTP_COOKIE	客戶機內的COOKIE 內容。
HTTP_USER_AGENT	提供包含了版本數或其他專有數據的客戶瀏覽器信息。
PATH_INFO	這個環境變量的值表示緊接在CGI程序名之後的其他路徑信息。它常常作為CGI程序的參數出現。

變量名	描述
QUERY_STRING	如果服務器與CGI程序信息的傳遞方式是GET，這個環境變量的值即使所傳遞的信息。這個信息經跟在CGI程序名的後面，兩者中間用一個問號'?'分隔。
REMOTE_ADDR	這個環境變量的值是發送請求的客戶機的IP地址，例如上面的192.168.1.67。這個值總是存在的。而且它是Web客戶機需要提供給Web服務器的唯一標識，可以在CGI程序中用它來區分不同的Web客戶機。
REMOTE_HOST	這個環境變量的值包含發送CGI請求的客戶機的主機名。如果不支持你想查詢，則無需定義此環境變量。
REQUEST_METHOD	提供腳本被調用的方法。對於使用HTTP/1.0 協議的腳本，僅GET 和POST 有意義。
SCRIPT_FILENAME	CGI腳本的完整路徑
SCRIPT_NAME	CGI腳本的的名稱
SERVER_NAME	這是你的WEB 服務器的主機名、別名或IP地址。
SERVER_SOFTWARE	這個環境變量的值包含了調用CGI程序的HTTP服務器的名稱和版本號。例如，上面的值為Apache/2.2.14(Unix)

每當客戶請求CGI 的時候，WEB服務器就請求操作系統生成一個新的CGI解釋器進程（如php-cgi.exe），CGI 的一個進程則處理完一個請求後退出，下一個請求來時再創建新進程。

當然，這樣在訪問量很少沒有並發的情況也行。但當訪問量增大，並發存在，這種方式就不適合了，於是就有了FastCGI

FastCGI

FASTCGI 是Web 服務器（ex:Nginx）和語言解釋器（ex:uWsgi）兩者底層的通信協議的規範，是對CGI的開放的擴展。

CGI的一個擴展，像是一個常駐（long-live）型的CGI ，廢除了CGI fork-and-execute （來一個請求fork 一個新進程處理，處理完再把進程kill 掉）的工作方式，轉而使用一種長生存期的方法，減少了進程消耗，提升了性能。

而FastCGI 則會先fork 一個master 進程，解析配置文件，初始化執行環境，然後再fork 多個worker 進程（與Nginx 有點像），當HTTP 請求過來時，master 進程將其會傳遞給一個worker 進程，然後立即可以接受下一個請求，這樣就避免了重複的初始化操作，效率自然也就提高了。

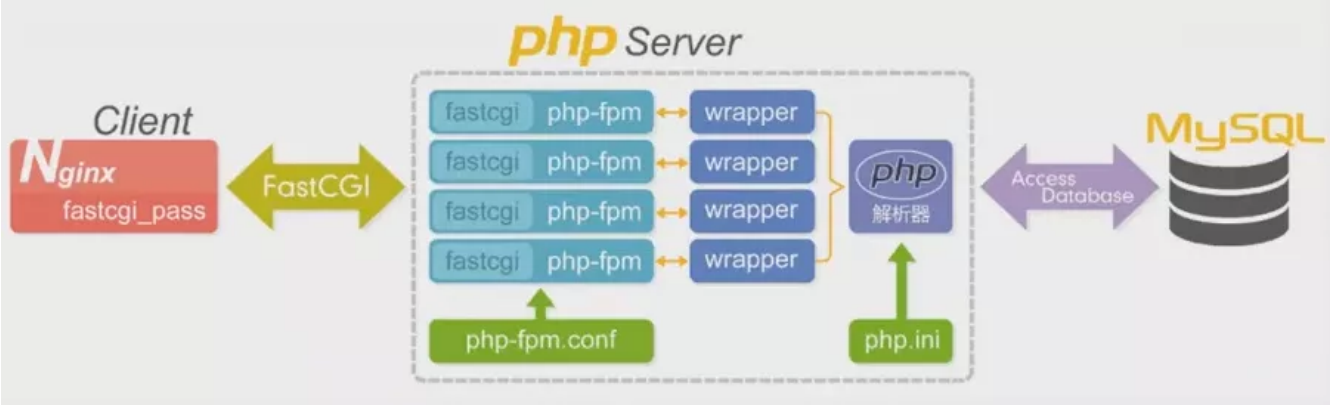
而且當worker 進程不夠用時，master 進程還可以根據配置預先啟動幾個worker 進程等著；當空閒worker 進程太多時，也會關掉一些，這樣不僅提高了性能，還節約了系統資源

php-fpm

FastCGI 只是一個協議規範，需要每個語言具體去實現，**PHP-FPM 就是PHP 版本的FastCGI 協議實現**，有了它，就是實現PHP 腳本與Web 服務器（通常是Nginx）之間的通信，同時它也是一個PHP SAPI，從而構建起PHP 解釋器與Web 服務器之間的橋樑。

Php-fpm全稱是php fastcgi process manager即php fastcgi進程管理器，相比fastcgi靜態的喚起cgi，fpm能根據訪問的壓力動態的喚起cgi進程和銷毀以到達動態的調整cgi數量，這樣可以有效的使用內存。

除此之外還有其它的一些優點，比如，fpm還可以平滑的重載php配置；由於fpm是使用Unix-Socket來和服務器通訊，所以也不用再配置cgi端口；fpm有更好的狀態輸出和slowlog日誌，502的時候能給出更多的錯誤細節。



PHP-FPM 負責管理一個進程池來處理來自Web 服務器的HTTP 動態請求，在PHP-FPM 中，master 進程負責與Web 服務器進行通信，接收HTTP 請求，再將請求轉發給worker 進程進行處

理，worker 進程主要負責動態執行PHP 代碼，處理完成後，將處理結果返回給Web 服務器，再由Web 服務器將結果發送給客戶端。這就是PHP-FPM 的基本工作原理

WSGI / uwsgi / uWSGI

在Python Web 開發中，我們經常使用Uwsgi 配合Nginx 部署一個Web 框架，如Django 或flask。同時我們又會說，框架和Web 服務器之間要符合WSGI 協議。

那就來釐清一下這幾個概念。

Web 服務器和Web框架

在講uWSGI 和WSGI 之前，先要弄清楚Web 開發的兩大塊，Web服務器和Web框架。

Web服務器即用來接受客戶端請求，建立連接，轉發響應的程序。至於轉發的內容是什麼，交由Web框架來處理，即處理這些業務邏輯。如查詢數據庫、生成實時信息等。Nginx就是一個Web服務器，Django或flask就是Web框架。

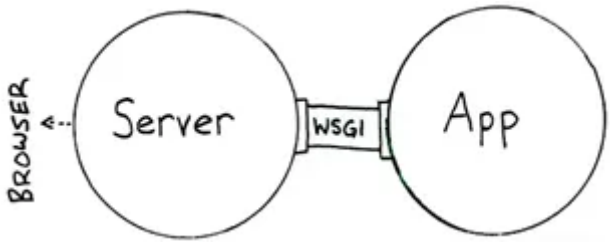
那麼如何實現uWSGI和WSGI的配合呢？如何做到任意一個Web服務器，都能搭配任意一個框架呢？這就產生了WSGI協議。只要Web服務器和Web框架滿足WSGI協議，它們就能相互搭配。所以WSGI只是一個協議，一個約定。而不是Python的模塊、框架等具體的功能。

而uWSGI，則是實現了WSGI協議的一個Web服務器。即用來接受客戶端請求，轉發響應的程序。實際上，一個uWSGI的Web服務器，再加上Django這樣的Web框架，就已經可以實現網站的功能了。

WSGI

WSGI，（WEB SERVER GATEWAY INTERFACE），Web服務器網關接口，是一種Web服務器網關接口，它是一個Web服務器（如Nginx，uWSGI等服務器）與Web應用（如Flask框架寫的程序）通信的一種規範。**當前運行在WSGI協議之上的Web框架有Bottle，Flask，Django**

實現了Python Web程序與服務器之間交互的通用性。有了這個東西，web.py或者bottle或者django等等的Python Web開發框架，就可以輕鬆地部署在不同的Web server上了，不需要做任何特殊配置（也需要一些小小的配置調整）



WSGI協議其實是定義了一種server與application解耦的規範，即可以有多個實現WSGI server的服務器，也可以有多個實現WSGI application的框架，那麼就可以選擇任意的server和application組合實現自己的Web應用。

例如uWSGI和Gunicorn都是實現了WSGI server協議的服務器，Django，Flask是實現了WSGI application協議的Web框架，可以根據項目實際情況搭配使用。

像Django，Flask框架都有自己實現的簡單的WSGI server，一般用於服務器調試，生產環境下建議用其他WSGI server，WSGI服務器的選擇很多，包括uWSGI和gunicorn

uwsgi

同WSGI一樣是一種通信協議

uwsgi協議是一個uWSGI服務器自有的協議，它用於定義傳輸信息的類型（type of information），每一個uwsgi packet前4byte為傳輸信息類型描述，它與WSGI相比是兩樣東西。

uWSGI (服務器)

它是一個Web服務器，它實現了WSGI協議、uwsgi、http等協議。用於接收前端服務器轉發的動態請求並處理後發給Web 應用程序。

因為apache也好，Nginx也罷，它們自己都沒有解析動態語言如php的功能，而是分派給其他模塊來做，比如apache就可以說內置了php模塊，支持的非常爽，讓人感覺好像apache就支持php一樣。uwsgi實現了WSGI協議、uwsgi、http等協議。Nginx中HttpUwsgiModule的作用是與uWSGI服務器進行交換。

uWSGI是使用C編寫的，顯示了自有的uwsgi協議的Web服務器。它自帶豐富的組件，其中核心組件包含進程管理、監控、IPC等功能，實現應用服務器接口的請求插件支持多種語言和平台，比如WSGI、Rack、Lua WSAPI，網管組件實現了負載均衡、代理和理由功能

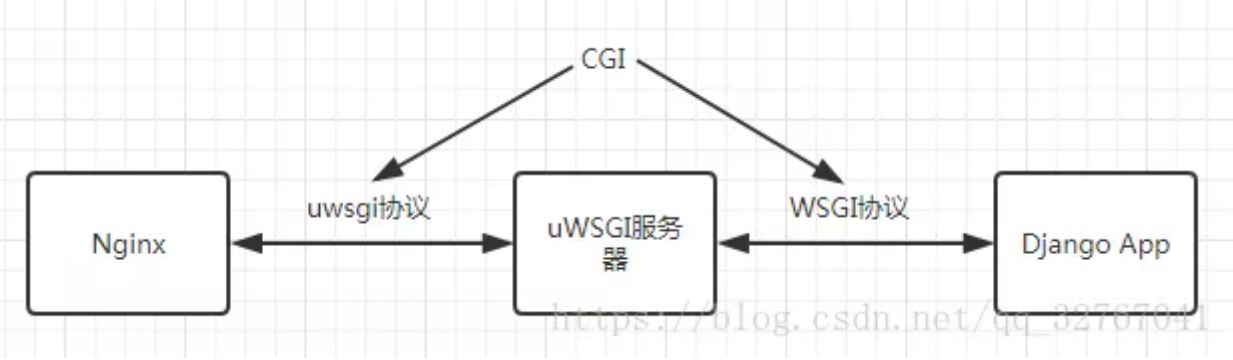
uWSGI也可以當做中間件。

- 如果是Nginx+uWSGI+App，那uWSGI就是一個中間件
- 如果是uWSGI+App，那它就是服務器

Nginx+uWGSI

假設我們使用Python 的Django 框架寫了一個網站，現在要將它掛在網上運行，我們一般需要：

- Nginx 做為代理服務器：負責靜態資源發送（js、css、圖片等）、動態請求轉發以及結果的回复。
- uWSGI 做為後端服務器：負責接收Nginx 轉發的請求並處理後發給Django 應用以及接收Django 應用返回信息轉發給Nginx。
- Django 應用收到請求後處理數據並渲染相應的返回頁面給uWSGI 服務器。

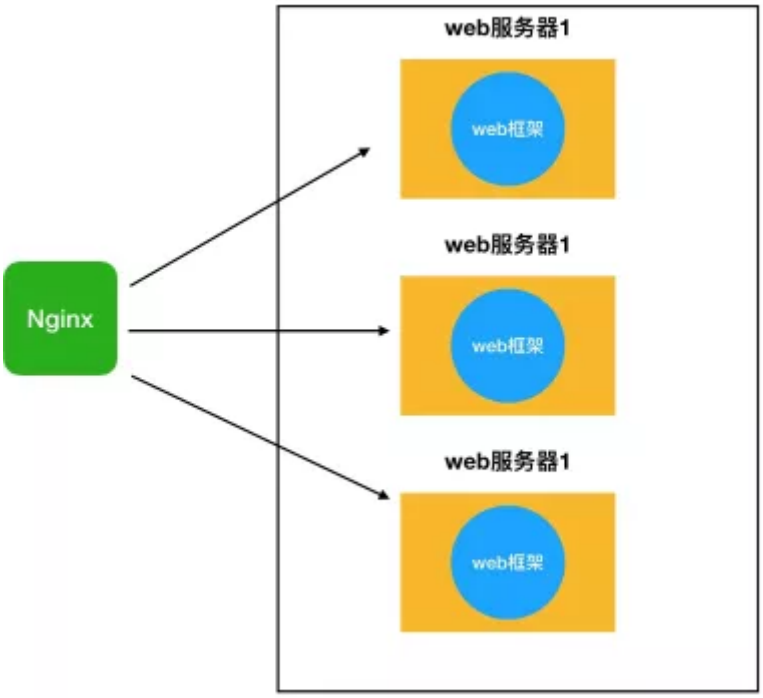


一個Django應用，通過WSGI協議連接uWSGI服務器，uWSGI服務器實現WSGI、http等協議，通過uwsgi協議和Nginx服務器實現http的動態請求和轉發以及結果

問題：有uWGSI 了，Django 為什麼還需要Nginx？

一個普通的個人網站，訪問量不大的話，當然可以由uWSGI 和Django 構成。但是一旦訪問量過大，客戶端請求連接就要進行長時間的等待。這個時候就出來了分佈式服務器，我們可以多來幾台Web 服務器，都能處理請求。

但是誰來分配客戶端的請求連接和Web 服務器呢？Nginx 就是這樣一個管家的存在，由它來分配。這也就是由Nginx 實現反向代理，即代理服務器。



Nginx 是一個HTTP 和反向代理服務器

- 正向代理：正向的就是由瀏覽器主動的想代理服務器發出請求，經代理服務器做出處理後再轉給目標服務器
- 反向代理：反向的就是不管瀏覽器同不同意，請求都會經過代理服務器處理再發給目標服務器

使用Nginx作為反向代理服務器的好處：

- 安全

不管什麼請求都要經過代理服務器，可以避免外部程序直接攻擊Web服務器

- 負載均衡

根據請求情況和服務器負載情況，將請求分配給不同的Web服務器，保證服務器性能

- 提高Web服務器的IO性能

請求從客戶端傳到Web服務器是需要時間的，傳遞多長時間就會讓這個進程阻塞多長時間，而通過反向代理，就可以由反向代理完整接受該請求，然後再傳給Web服務器，從而保證服務器性能，而且有的一些簡單的事情（比如靜態文件）可以直接由反向代理處理，不經過Web服務器

總結

- WSGI是一種通信協議
- uwsgi是一種通信協議，常用於在uWSGI服務器與其他網絡服務器的數據通信

- 而uWSGI是實現了uwsgi和WSGI兩種協議的Web服務器

百度百科上說uwsgi是一種線路協議而不是通信協議，個人更傾向於uwsgi是類似WSGI的通信協議的說法，uwsgi和WSGI都是基於CGI擴展出來的。

ASGI

異步網關協議接口，一個介於網絡協議服務和Python應用之間的標準接口，能夠處理多種通用的協議類型，包括HTTP，HTTP2和WebSocket。

然而目前的常用的WSGI主要是針對HTTP風格的請求響應模型做的設計，並且越來越多的不遵循這種模式的協議逐漸成為Web變成的標準之一，例如WebSocket。

ASGI嘗試保持在一個簡單的應用接口的前提下，提供允許數據能夠在任意的時候、被任意應用進程發送和接受的抽象。並且同樣描述了一個新的，兼容HTTP請求響應以及WebSocket數據幀的序列格式。允許這些協議能通過網絡或本地socket進行傳輸，以及讓不同的協議被分配到不同的進程中。

WSGI和ASGI的區別

WSGI是基於HTTP協議模式的，不支持WebSocket，而ASGI的誕生則是為了解決Python常用的WSGI不支持當前Web開發中的一些新的協議標準。同時，ASGI對於WSGI原有的模式的支持和WebSocket的擴展，即ASGI是WSGI的擴展。

參考

https://www.cnblogs.com/wanghetao/p/3934350.html
https://baike.baidu.com/item/fastcgi/10880685
https://www.jianshu.com/p/679dee0a4193
https://baijiahao.baidu.com/s?id=1590941335729952485&wfr=spider&for=pc
https://blog.csdn.net/qq_35318838/article/details/61198183

往期內容：

Python 圖形界面框架PyQt5 使用指南！

再見，Teamviewer！這款國產輕量級遠程桌面軟件超牛逼！

我在美團的八年

厲害了！推荐一個Web 端自動化神器- Automa

深耕Python快10年了，積累了不少好書，準備了一份Python經典電子書給廣大粉絲。大家有需要的添加下方微信，暗號「**p y資料**」，我私發給大家哈

暗號「py資料」獲取哦

喜歡此內容的人還喜歡

Python 蟬聯榜首，PHP 即將跌出前十| TIOBE 11 月編程語言排行榜

Python那些事

牢記“四不要”，寫好Python 的Lambda 函數

Python那些事

Python的打包神器——Nuitka

21CTO