
Tornado

Cos’è Tornado

Tornado è un framework web ed un insieme di librerie per il networking. Utilizza protocolli di comunicazione di rete non bloccanti ed è quindi capace di scalare a diverse decine di migliaia di connessioni contemporanee.

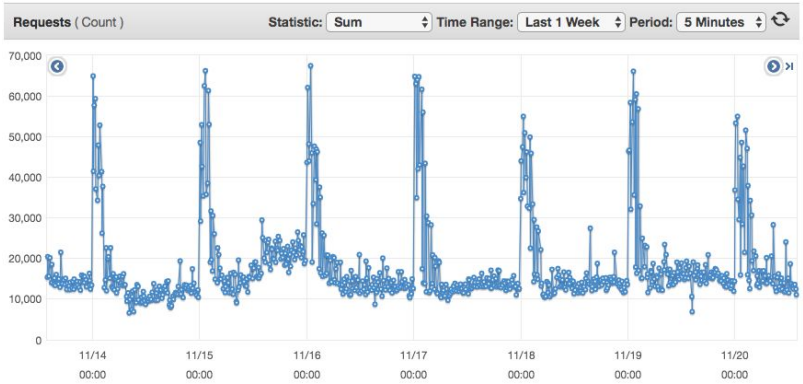
Casi d’uso

- Applicazioni asincrone
- Microservizi RESTful
- Websocket
- Connessioni lunghe o persistenti

Installazione

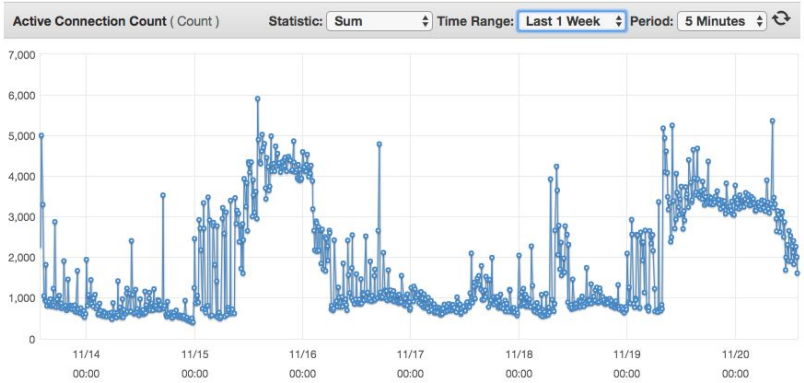
```
> pip install tornado
```

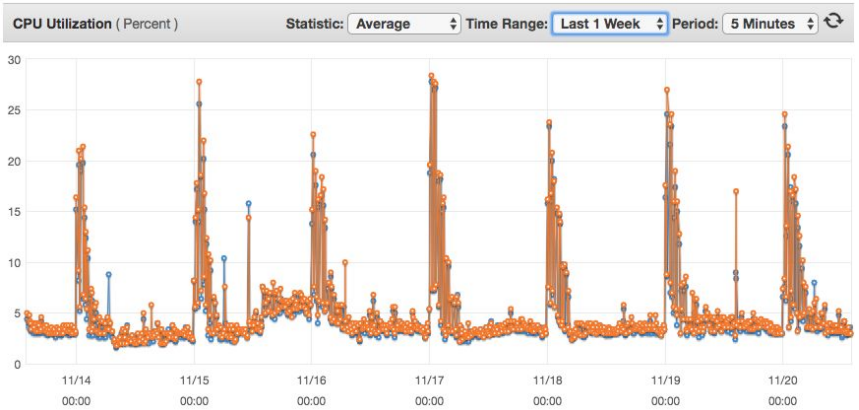
Tornado nella vita reale



Server m5.large AWS
70$/mese
2 istanze
2 x Xeon® Platinum 8175 da 2,5 GHz
8 GB Ram

evensi™
events for me





Codice

import tornado.ioloop

E’ la classe che gestisce la connessione tramite socket con logica non bloccante. (https://www.tornadoweb.org/en/stable/ioloop.html)

import tornado.web

Si occupa di fornire un set di funzioni asincrone per gestire le richieste http. (https://www.tornadoweb.org/en/stable/web.html)

class MainHandler(tornado.web.RequestHandler)

Si occuperà di gestire la richiesta che arriva su un determinato endpoint (in questo caso la root /). Eredita la classe RequestHandler ed implementa i vari metodi get(), post(), delete(), ecc dell’http.

def make_app()

Definisce la struttura dell’applicazione Tornado, quali endpoint gestisce e quali classi vengono attribuite agli endpoint. Le classi gestiranno poi le chiamate che arrivano sugli endpoint.

WebSocket

- Crea un canale di comunicazione su una singola connessione TCP
- E’ pienamente compatibile con il protocollo HTTP
- E’ utilizzato per comunicazioni che necessitano uno scambio dati real-time
- Connessione sempre presente
- Il server può mandare dati al client senza che questo li richieda esplicitamente
- Di default apre le connessioni sulle porte dell’HTTP/S (80, 443)
- Compatibile con tutti i maggiori browser

WebSocket in Tornado

Classe: tornado.websocket.WebSocketHandler()
Metodi più importanti della classe:

open(): invocato all’apertura della connessione
on_message(): invocato quando qualcuno invia messaggi nella connessione verso il server
on_close(): invocato alla chiusura della connessione
write_message(): scrive un messaggio verso il client sulla connessione

Extras

asyncio framework

- E’ un insieme di funzionalità per la programmazione asincrona basata sulle coroutines
- Le coroutines sono funzioni che si possono “mettere in pausa”.
- Utilizza le parole chiave “async” per definire una funzione asincrona ed “await” per contrassegnare la funzione da attendere prima di procedere con il resto del corpo della funzione
- E’ un modo per mettere in pausa l’esecuzione di una funzione, nell’attesa di un risultato da una chiamata
- Fa uso dell’*event loop* per eseguire le coroutines

Articoli utili:
https://snarky.ca/how-the-heck-does-async-await-work-in-python-3-5/

https://medium.com/@jialun.tom.chen/using-python-async-await-to-get-your-data-fast-fe6d9ce7c6e