Python: misurare le prestazioni di un sito web

Misurare le metriche di risposta di un indirizzo web è essenziale per comprendere le prestazioni e la disponibilità di un sito. In questo articolo, esploreremo come utilizzare Python per raccogliere dati dettagliati sulle prestazioni di un sito web, comprese la latenza, la disponibilità e altre metriche chiave. Utilizzeremo diverse librerie Python per eseguire queste operazioni, tra cui `requests`, `time`, `statistics`, e `matplotlib` per la visualizzazione dei dati.

Prima di iniziare, assicurati di avere installato Python e le seguenti librerie:

```
pip install requests matplotlib
```

Per misurare la latenza della risposta di un sito web, possiamo utilizzare la libreria `requests` per inviare una richiesta HTTP e misurare il tempo impiegato per ottenere una risposta.

```
import requests
import time

def measure_latency(url, num_requests=10):
    latencies = []
    for _ in range(num_requests):
        start_time = time.time()
        response = requests.get(url)
        latency = time.time() - start_time
```

```
        latencies.append(latency)
        print(f'Response status:
{response.status_code}, Latency: {latency:.4f}
seconds')
    return latencies

url = "https://www.example.com"
latencies = measure_latency(url)
```

Dopo aver raccolto i dati sulla latenza, possiamo analizzarli per ottenere informazioni utili come la media, la deviazione standard e i percentili.

```
import statistics

def analyze_latencies(latencies):
    mean_latency = statistics.mean(latencies)
    median_latency = statistics.median(latencies)
    stdev_latency = statistics.stdev(latencies)
    percentile_90_latency =
statistics.quantiles(latencies, n=100)[89]

    print(f'Mean latency: {mean_latency:.4f}
seconds')
    print(f'Median latency: {median_latency:.4f}
seconds')
    print(f'Standard deviation: {stdev_latency:.4f}
seconds')
    print(f'90th percentile latency:
{percentile_90_latency:.4f} seconds')

analyze_latencies(latencies)
```

Per una migliore comprensione delle prestazioni del sito web, è utile visualizzare i dati raccolti. Utilizzeremo `matplotlib` per creare grafici.

```
import matplotlib.pyplot as plt

def plot_latencies(latencies):
    plt.figure(figsize=(10, 6))
    plt.plot(latencies, marker='o')
    plt.title('Website Latency Over Time')
    plt.xlabel('Request Number')
    plt.ylabel('Latency (seconds)')
    plt.grid(True)
    plt.show()

plot_latencies(latencies)
```

Oltre alla latenza, è importante monitorare la disponibilità del sito web. Possiamo farlo registrando lo stato delle risposte HTTP.

```
def monitor_availability(url, num_requests=10):
    availability = []
    for _ in range(num_requests):
        response = requests.get(url)
        availability.append(response.status_code ==
200)
        print(f'Response status:
{response.status_code}')
    uptime = sum(availability) / num_requests * 100
    print(f'Uptime: {uptime:.2f}%')
```

```
monitor_availability(url)
```

Conclusione

Misurare le metriche di risposta di un indirizzo web è un compito fondamentale per garantire che il sito web offra una buona esperienza utente. Utilizzando Python e le librerie `requests`, `time`, `statistics` e `matplotlib`, possiamo raccogliere, analizzare e visualizzare i dati di prestazione del sito web in modo efficace. Questo ci permette di identificare e risolvere eventuali problemi di prestazioni, migliorando la disponibilità e la velocità del sito.

Seguendo i passaggi descritti in questo articolo, sarai in grado di implementare un sistema di monitoraggio delle prestazioni per qualsiasi sito web, ottenendo così una visione dettagliata delle sue metriche di risposta.