RightSizing-SpikeServer

1 Caso di Studio

Il sistema oggetto di studio è un'architettura di data center per un Internet Service Provider, progettata per gestire dinamicamente le fluttuazioni di carico e garantire la Quality of Service (QoS) ottimizzando al contempo l'uso delle risorse.

Il problema principale affrontato è il "right-sizing", ovvero come evitare sia il sovradimensionamento (spreco di risorse) sia il sottodimensionamento (violazione degli SLA e degrado delle prestazioni), specialmente in presenza di fluttuazioni di carico a breve e lungo termine.

L'architettura proposta, come descritto nel caso di studio 6.2 del libro di testo "Performance Engineer", si basa su un livello di scaling verticale che gestisce i picchi di carico improvvisi e di breve durata. Questo livello introduce uno Spike Server dedicato. Un Load Controller monitora un indicatore di picco (Spike Indicator, SI), definito come il numero di richieste concorrenti totali presenti su un Web Server (tutte le richieste vengono gestite con un scheduler processor sharing).

Il comportamento del sistema seguirebbe quanto descritto:

- Quando l'indicatore SI supera una soglia di allarme SI_max, le nuove richieste in arrivo non vengono più inviate al Web Server congestionato, ma vengono reindirizzate allo Spike Server.
- Quando il carico sul Web Server diminuisce e SI scende al di sotto della soglia, il routing delle richieste torna alla normalità.

2 Obiettivi dello studio

Lo studio si pone l'obiettivo di analizzare e validare l'efficacia del modello di autoscaling basato su spike server attraverso la simulazione. Gli obiettivi specifici sono:

- Determinare il valore di SI_max più alto possibile (provando vari valori di SI) che mantenga comunque il tempo di risposta medio

$$E[R] \leq 8$$

secondi, con un tasso di arrivo iperesponenziale con media di 6.66 req/s (400 req/min).

- Analizzare come varia il tempo di risposta medio al variare del coefficiente di variazione degli arrivi iperesponenziali (fluttuazioni a breve termine) per carichi di lavoro crescenti (da 1 req/s a 12 req/s).
- Verificare come cambia il contesto (sempre sotto carichi crescenti e al variare di SI_max) se lo spike server ha un tasso di servizio doppio rispetto al webserver principale, invece che uguale come nel caso base.

3 Modello concettuale

Il modello descritto può essere schematizzato come in figura 1.

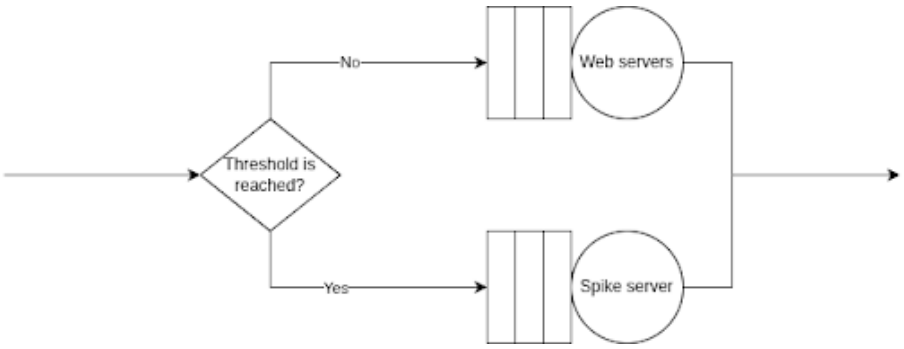

Figure 1: Modello concettuale del sistema di autoscaling gerarchico con Spike Server

I job arrivano e a seconda del livello di pienezza dei webservers. Nonostante il sistema possa sembrare troppo semplice per una analisi simulativa, il fatto che il routing non sia probalistico lo rende molto complesso da analizzare in modo analitico. il routing dei job, dipende strettamente dallo stato del webserver nel momento del routing.

Spiegazione tempi di servizio esponenziale: Nel caso di studio affrontato nel libro viene utilizzata una distribuzione iperesponenziale nel tassi di servizio per modellare il fatto che ad un server arrivano job di dimensione molto variabile. Ovviamente questa cosa potrebbe essere modellata anche utilizzando delle size dei job differenti anziché agire sul tasso di servizio. Tuttavia ho deciso di attenermi al testo originale e utilizzare anche io dei tassi di servizio iperesponenziali per modellare questo comportamento per poter confrontare meglio i risultati ottenuti.

3.1 Spike server

Lo Spike Server avrà la stessa potenza rispetto al webserver nei primi 2 obiettivi. Nell'obiettivo 3, invece, avrà un tasso di servizio doppio rispetto a quello del webserver principale, in modo da poter smaltire rapidamente i picchi di carico. Per il calcolo di SI_max ottimle è importante sottolineare che lo spike server è bene usarlo il meno possibile, in quanto è una risorsa più costosa rispetto al webserver. Quindi anziché cercare di minimizzare il tempo di risposta medio del sistema, si cerca di rispettare lo SLA (8 secondi) utilizzando il meno possibile lo spike server.

3.2 Web Server

Il web server sarà solamente uno, per semplicità di modellazione. Per questo motivo non vengono gestite le fluttuazioni a lungo termine, in quanto queste richiederebbero un sistema con più webserver per utilizzare una scaling orizzontale.

4 Modello delle specifiche

Queste variabili matematiche, sono la rappresentazione dello stato del sistema:

- $SI(t)$: Spike indicator, indica il numero di job presenti nel web server al tempo t.
- $n_spike(t)$: numero di job presenti nello spike server al tempo t.

4.1 Componenti fisici del sistema:

- Web server principale: server principale che gestisce le richieste in arrivo.
- Spike server: server secondario che viene attivato quando il carico sul web server principale supera una certa soglia (SI_max).

Componenti logici del sistema:

- Load Controller: componente che monitora l'indicatore di picco (Spike Indicator, SI) e decide il routing delle richieste in base al valore di SI e alla soglia SI_max.

4.2 Specifiche del carico di lavoro:

- Arrivi:
  - Valore base: Processo di arrivo iperesponenziale con cv = 4 e media 0.15 secondi (6.66 req/s)
  - Valore stress: Processo di arrivo iperesponenziale con carico variabile da 1 req/s a 12 req/s

- Servizio: Distribuzione iperesponenziale con cv = 4 e media 0.16 secondi (web server e spike server), media 0.08 secondi (spike server nell'obiettivo 3)
- Scheduling: Processor Sharing

Vale la pena notare che, nel caso base del webserver, siccome $\rho > 1$, il sistema è instabile e senza spike server il tempo di risposta divergerebbe a infinito.

4.3 Logica di controllo:

- Arrivi nuovi job:
  - Se $SI < SI\_max$: invia al webserver principale
  - Se $SI \geq SI\_max$: invia allo spike server
- Completamento job:
  - Decrementa il contatore SI se il job era nel webserver principale
  - Decrementa il contatore n_spike se il job era nello spike server

Nella logica originale del caso di studio, il valore di SI parte da SI_max e viene decrementato ad ogni completamento di un job. La logica da me adottata, nonostante sia inversa, è perfettamente equivalente.

4.4 Metrica di valutazione delle prestazioni:

Lo SLA da rispettare è un tempo di risposta medio $E[R] \leq 8$ secondi.

4.5 Gestione della fine della simulazione:

Dato che il sistema viene studiato a regime, esso fa riferimento ad un orizzonte temporale prefissato che va dai 120 secondi (per eliminare il bias della fase transitoria in cui le code sono ancora vuote) ai 1200 secondi di simulazione. Questo viene fatto nel report di riferimento e quindi per confrontare al meglio i risultati si è deciso di adottare lo stesso approccio. Inoltre, non verrà fatta allo scadere del tempo di simulazione una pulizia delle code, ma si considereranno solo i job completati entro il tempo di simulazione, per il calcolo delle metriche medie aggregate (es. R0), coerentemente con l'approccio adottato nel caso di studio di riferimento.

5 Modello Computazionale

5.1 Stato del sistema

- SI: Spike indicator (numero di job nel web server)
- n_spike: numero di job nello spike server

5.2 Identificazione degli eventi

- arrivo di un nuovo job
- completamento di un job nel web server
- completamento di un job nello spike server

5.3 Logica di controllo e routing

Ogni volta che arriva un nuovo job:

- Se $SI \leq SI\_max$, viene incrementato SI di 1 e il job viene inviato al web server.
- Se $SI > SI\_max$, viene incrementato n_spike di 1 e il job viene inviato allo spike server.

Ogni volta che job termina decremenenta il valore di SI.

5.4 Implementazione dello scheduler

Al contrario di quanto avviene nel caso di studio di riferimento in cui si usa uno scheduling processor sharing.

5.5 Configurazione dei parametri delle distribuzioni

- Stream 0: Inter-arrivi (Iperesponenziale, media 0.15s, cv=4).
- Stream 1: Servizio Web Server (Iperesponenziale, media 0.16s, cv=4).
- Stream 2: Servizio Spike Server (Iperesponenziale, media 0.08s, cv=4).