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 gerarchico attraverso la simulazione. Gli obiettivi specifici sono:

- identificare il valore di carico di lavoro che supera i tempi di risposta richiesti
- Identificare il valore ottimale di SI_max: Per un dato carico di lavoro (es. 40,000 richieste/ora), determinare il valore di SI_max che minimizza il tempo di risposta medio del sistema o lo mantiene al di sotto di un valore

target definito da un Service Level Agreement, come gli 8 secondi utilizzati nel libro.

- Studiare il comportamento del sistema sotto carichi crescenti: Analizzare le prestazioni del sistema (in particolare il tempo di risposta) al variare del tasso di arrivo delle richieste (carico leggero, medio, pesante) utilizzando la soglia SI_max.
- Studiare il sistema con fluttuazioni a breve termine: Analizzare le prestazioni del sistema introducendo fluttuazioni a breve termine, utilizzando sempre la soglia SI_max
- Trovare due nuovi SI_max per fluttuazioni a breve termine per rimanere sotto un certo tempo di risposta (8 secondi)

2.1 Obiettivi 2

Gli obiettivi specifici sono:

- **Valutazione della capacità nominale**: Identificare il limite critico del carico di lavoro (tasso di arrivo) che porta alla violazione dei tempi di risposta definiti dagli SLA in assenza di meccanismi di scaling.
- **Ottimizzazione della soglia SImax**: Determinare il valore ottimale dello *Spike Indicator* (SI) — inteso come numero totale di job nel sistema (in coda e in servizio) — che minimizzi il tempo di risposta medio o lo mantenga entro il target di 8 secondi per un carico di lavoro di riferimento (es. 40.000 richieste/ora).
- **Analisi di sensibilità sotto carichi crescenti**: Studiare le prestazioni del sistema al variare del tasso di arrivo per testare la stabilità della soglia SImax in scenari di carico leggero, medio e pesante.
- **Analisi della robustezza ai picchi (*Burstiness*)**: Valutare l'efficacia dello *Spike Server* (risorsa tipicamente più performante del Web Server principale) nel gestire fluttuazioni di carico a breve termine modellate tramite distribuzioni iperesponenziali.
- **Ottimizzazione dinamica per scenari di picco**: Identificare valori specifici di SImax capaci di garantire il rispetto degli SLA anche in presenza di diverse intensità di picco improvviso, confrontando l'efficacia della soglia fissa rispetto a scenari di carico variabile.

2.2 Obiettivi 3

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 40,000 richieste/ora.

- Analizzare come varia il tempo di risposta medio al variare del coefficiente di variazione degli arrivi iperesponenziali (fluttuazioni a breve termine) mantenendo il tasso di arrivo medio a 40,000 richieste/ora e utilizzando il valore di SI_max trovato nel punto precedente.

- Verificare come cambia il contesto se lo spike server ha un tasso di servizio triplo rispetto al webserver principale, invece che doppio come nel caso base.

3 Modello concettuale

Il modello descritto può essere schematizzato come in figura 1.

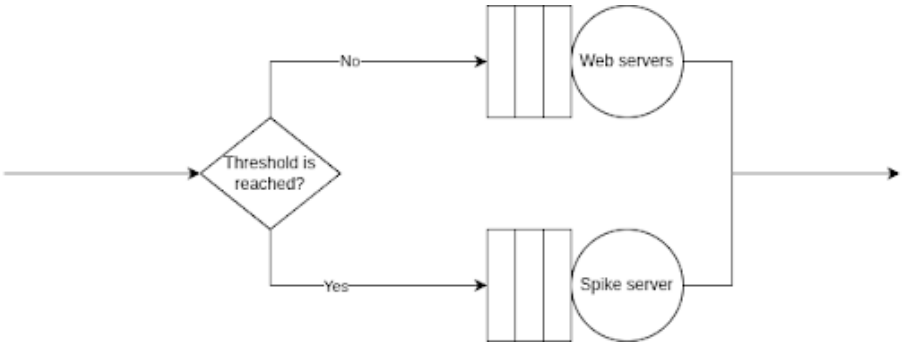

Figure 1: Modello concettuale del sistema di autoscaling gerarchico con Spike Server

I job arrivano e a seconda del livello di pienezza dei webservers. Nonostante il sistema possa sembrare troppo semplice per una analisi simulativa, ci sono una serie di aspetti che lo rendono complesso e difficilmente modellabile solo matematicamente senza semplificazioni:

- Il routing non è probabilistico: il routing dei job non è semplicemente probabilistico (40% su uno e 60% su un altro), ma dipende strettamente dallo stato del webserver nel momento del routing. Questa complicazione rende molto difficile un'analisi statica, soprattutto nel transiente.

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.

3.1 Spike server

Lo Spike Server avrà una potenza maggiore rispetto al webserver. In particolare 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 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

Componenti fisici del sistema:

- Web server principale
- Spike server

Componenti logici del sistema:

- Load Controller

Specifiche del carico di lavoro:

- Arrivi: Processo di arrivo iperesponenziale con cv = 4 e media 0.15 secondi (40,000 richieste/ora)
- Servizio: Distribuzione iperesponenziale con cv = 4 e media 0.16 secondi (web server), media 0.08 secondi (spike server)
- Scheduling: Processor Sharing

Classi di clienti:

- Arrivi normali

- Token: sono gettoni di controllo che servono al Load Controller. C'è un numero fisso di token (SI_max). Ogni volta che arriva una richiesta, se ci sono token disponibili, ne viene consumato uno e la richiesta viene inviata al webserver principale. Se non ci sono token disponibili, la richiesta viene inviata allo spike server. Quando una richiesta termina il servizio, il token viene restituito al Load Controller.

Logica di controllo:

- Se $SI < SI\_max$: invia al webserver principale
- Se $SI \geq SI\_max$: invia allo spike server