Test plan:

- thread group

insieme di utenti che fanno delle richieste, posso settare il numero di clienti, il ramp-up period (tempo di avvio di tutti i thread, per esempio 1 ogni secondo x 10 volte), il numero di volte per cui un thread fa una richiesta

ps: non voglio un comportamento deterministico delle richieste, visto che ogni utente si comporta in modo diverso => uso lo scheduler e la durata in secondi (durata del test)

- sampler

richieste, possono essere di tipo HTTP.

request default, utile quando mi scoccio di inserire il server name o l'ip address del server, così lo definisco per default

- logic controller

ci permettono di dare una logica all'esperimento, random controller permette di fare una richiesta in modo random ad un thread (la risorsa è random), il random order controller ci permette di chiedere tutte le risorse in modo random (l'ordine è random)

il controller deve vedere le richieste http QUINDI le richieste devono essere aggiunte sotto il controllore

- timer

constant throughput timer, fa in modo che un thread o più thread raggiungano un target throughput, in termini di richieste per minuto (requisito che può soddisfare un singolo thread oppure un gruppo di thread insieme)

- listener

ci permettono di collezionare varie metriche, il problema è che consumano troppa memoria se ce ne sono troppi

Collezionamento dei parametri

parametri collezionabili sia lato client che lato server. Lato client può essere la percentuale di errore, throughput o tempo di risposta.

Utilizziamo il listener simple data writer (gli altri consumano un sacco di memoria), salva i risultati su un file, posso configurare le metriche da collezionare:

- timestamp: quando fatta la richiesta
- nome thread
- page: che tipo di pagina è stata richiesta
- byte count: dimensione della risorsa richiesta
- latency, success/errors, tempo trascorso(elapsed time) per analisi di performance: latency è il tempo tra il client che fa la richiesta e il server che prende in carico la richiesta, elasped time è il tempo che io client faccio la richiesta e il server me la ritorna (la latency è compresa in questo tempo)

vmstat --n 1 10, collezionare per 10 secondi con periodo di campionamento 1 secondo informazioni sul server, mi dà informazioni su memoria, i/o, cpu, ecc.., posso farlo partire per 300 sec e poi analizzo i parametri lato server

Capacity test

capacity test: caratterizzare le performance di un sistema al limite, lo faremo attraverso alcune metriche collezionate lato client

utile per la capacity planning

throughput: il tasso per cui le richieste possono essere servite dal sistema,

capacità nominale: massima capacità ottenibile dal sistema in condizioni ideali, non riuscirò mai ad arrivarci con degli esperimenti perchè è un modello matematico

knee capacity: punto di funzionamento ottimo

se c'è uno sbilanciamento del carico, dalla usable capacity posso ottenere una decrescita del throughput. in termini di reliabilty nel punto di usable capacity non mi conviene lavorare mai (far lavorare un motore ad alte velocità SEMPRE non mi conviene), in termini di availabilty invece va ancora bene perchè il server funziona ancora se perde una risorsa, il mean time to repair è basso

reliability: continuità del servizio

availability: se il sistema si rimette subito in piedi, il tempo di recovery è basso, è molto available anche se fallisce spesso

come si fa il capacity test: (l'obbiettivo è costruire le due curve)

- analizzare throughput e tempo di risposta (elapsed time) incrementando i valori del carico utilizzando il constant throughput timer e mantenendo costante la configurazione del test plan (a parte che per il carico del CTT)
- fare le ripetizioni per ogni valore dell'asse delle x (il carico), uso la media o la mediana di questi valori
- costruisco il grafico
- trovo il knee e l'usable capacity, per trovare il knee posso anche costruire il grafico di power e trovarne il massimo

esercizio: non faccio misurazioni troppo fitte altrimenti ci vuole troppo tempo

30-50 thread

loop count infinito

esercizio2: faccio diversi thread group, con ognuno un CTT diverso, per ogni thread group aggiungo un summary report (è un listener) così posso vedere per ognuno il throughput