Gestione di un Servizio Cloud

Analisi, modellazione, simulazione e valutazione di un sistema

Gabriele Quatrana 0306403

AA 2021/2022

Indice

1 Introduzione

Questo documento descrive l'analisi delle performance di un **Servizio Cloud** che permette di calcolare operazioni matematiche di carattere generale attraverso l'architettura remota messa a disposizione degli utenti.

Per permettere l'accesso al servizio, è necessario controllare che l'utente abbia un account valido. Gli utenti che passano il controllo possono essere di due tipi:

- Abbonati: la loro richiesta viene inoltrata a un nodo dedicato che fornisce maggiori prestazioni.
- Non abbonati: la loro richiesta viene inoltrata a un nodo che fornisce prestazioni minori.

Dopo che le richieste vengono elaborate dai rispettivi centri, viene eseguito un controllo aggiuntivo sulla validità della risposta prodotta, per garantire un servizio ottimale.

2 Obiettivo

L'obiettivo di questo progetto è quello di minimizzare i costi di gestione del servizio individuando il numero ottimale di serventi per ogni nodo nelle diverse fasce orarie. In particolare, si vogliono rispettare due requisiti di QoS:

- Mantenere il tempo di risposta complessivo del sistema inferiore ai 25 secondi.
- Validare la risposta prodotta almeno per il 65% degli utenti che accedono al servizio.

Per raggiungere l'obiettivo prefissato, è stato eseguito uno uno studio sia dello stato stazionario che transiente del servizio, provando diverse configurazioni di serventi per individuare quale di queste rispetta i due vincoli mantenendo il minor costo totale.

3 Modello Concettuale

Il servizio è stato modellato secondo la seguente rete di nodi:

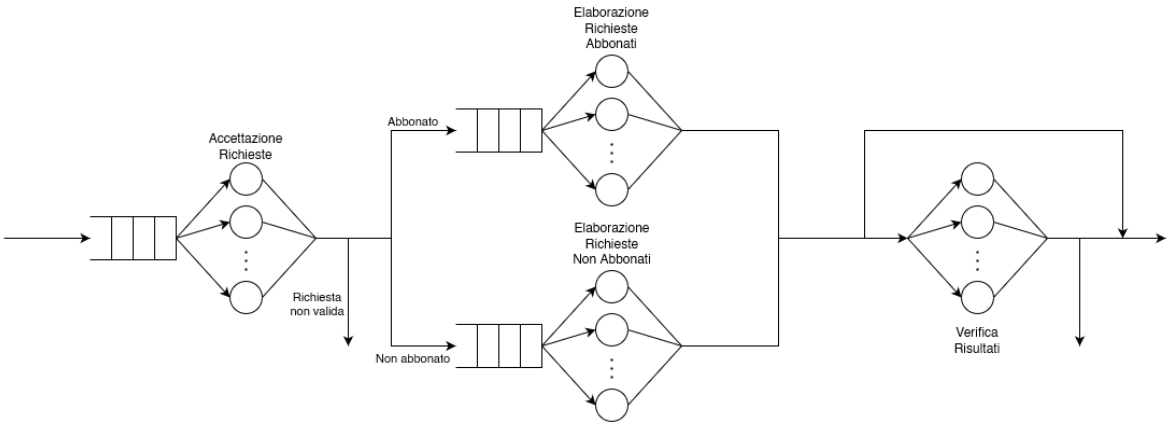


Figura 1: Rete di code per il servizio cloud.

Sono stati definiti quattro nodi differenti:

- Nodo 0: Accettazione Richieste.
- Nodo 1: Elaborazione Richieste Abbonati.
- Nodo 2: Elaborazione Richieste Non Abbonati.
- Nodo 3: Validazione Risposta.

I primi tre nodi sono stati modellati come una **M/M/k**: le code sono infinite in quanto il servizio non impone un limite sul numero di richieste che può servire. All'arrivo di un nuovo job nel nodo, si verifica se è presente un servente libero: se almeno uno di questi è libero, il job viene assegnato al primo di questi, altrimenti viene inserito nella coda del nodo.

L'ultimo nodo è stato modellato come una **M/M/k/k** (loss system): se tutti i server sono occupati gli utenti riceveranno direttamente la risposta senza la validazione di questa. Uno degli obiettivi dello studio è proprio quello di dimensionare la rete in modo tale che la percentuale di utenti per i quali non venga effettuato questo controllo sia al massimo del 35%.

Il servizio cloud è stato analizzato durante una giornata completa (24 ore). Supponiamo che il numero di utenti che utilizzano il servizio non sia costante, ma che ci siano periodi con maggiore carico. Per cui, sono state individuate tre fasce orarie:

- **02:00 - 09:00**: **7** ore con carico di lavoro basso.
- **09:00 - 21:00**: **12** ore con carico di lavoro intenso.
- **21:00 - 02:00**: **5** ore con carico di lavoro medio.

Le variabili di stato considerate sono le seguenti:

- Numero di serventi disponibili per ogni nodo nelle tre fasce orarie.
- Stato di un servente: libero/occupato, attivo/inattivo ed utilizzato/non utilizzato.

Gli utenti che utilizzano il servizio possono essere di due tipi:

- Abbonati: finanziano il servizio attraverso un pagamento mensile.
- Non abbonati: utilizzano il servizio in modalità gratuita.

Gli eventi che gestisce il sistema sono i seguenti:

- Arrivo di una richiesta da parte di un utente.
- Servizio di una richiesta presso l'apposito nodo in base alla tipologia di utente che l'ha effettuata.
- Cambio di fascia oraria.

Il cambio di fascia oraria può causare l'attivazione o la disattivazione di alcuni serventi per mantenere il sistema stabile e far fronte alle variazioni del flusso di ingresso.

Siccome ogni servente attivo comporta un maggiore costo di gestione, l'obiettivo è quello di trovare il minor numero di serventi, per ogni fascia oraria, che permetta di minimizzare le spese ma rispettando i due requisiti di QoS.

4 Modello Delle Specifiche

Per facilitare il calcolo dei vari parametri di input della simulazione è stato creato un foglio di calcolo *Excel*, in modo da avere un riscontro immediato nel caso in cui vengono cambiati tali parametri.

4.1 Dati in Input

Per la scelta dei parametri di input del servizio sono stati considerati i dati del servizio cloud di *Microsoft*, chiamato *Azure*. Secondo diverse analisi statistiche sul numero di utenti del servizio (`https://www.usesignhouse.com/blog/microsoft-azure-stats`), *Microsoft Azure* ha **722.22** milioni di utenti, di cui circa **270.000** attivi quotidianamente.

I tempi di interarrivo ed i tempi di servizio sono stati modellati secondo una distribuzione *Esponenziale*. Sono stati considerati i seguenti parametri di input per il servizio esaminato:

- Periodo di osservazione: **24** ore (86400 secondi).
- Numero di utenti al giorno: **270.000** utenti con **3,125** arrivi medi al secondo.

Gli arrivi sono stati suddivisi nelle tre fasce orarie individuate:

- 02:00 - 09:00: 15% degli utenti giornalieri, ovvero **40500** utenti con **1,607143** arrivi al secondo.
- 09:00 - 21:00: 65% degli utenti giornalieri, ovvero **175500** utenti con **4,0625** arrivi al secondo.
- 21:00 - 02:00: 20% degli utenti giornalieri, ovvero **54000** utenti con **3,00** arrivi al secondo.

Per i vari nodi sono stati definiti i seguenti tempi di servizio:

- Tempo di accettazione della richiesta: **1** secondo.
- Tempo di elaborazione della richiesta per utenti abbonati: **5** secondi.
- Tempo di elaborazione della richiesta per utenti non abbonati: **15** secondi.
- Tempo di validazione del risultato prodotto: **3** secondi.

Di seguito vengono indicate le percentuali di suddivisione degli utenti in base alle varie fasi del servizio:

- Percentuale di richieste non valide: **10%**.
  - Probabilità che una richiesta venga scartata: $p_{rejected} = 0.1$.
  - Probabilità che una richiesta venga accettata: $p_{accepted} = 0.9$.
- Percentuale di utenti abbonati: **32,4%**.
  - Probabilità che un utente sia abbonato: $p_{subscriber} = 0,324$.
  - Probabilità che un utente non sia abbonato: $p_{not\_subscriber} = 0,676$.

4.2 Probabilità di Routing

Il tipo di utente che accede al servizio non viene generato in modo aleatorio, ma il flusso di ingresso nel servizio viene distribuito tramite delle probabilità di routing.
Sono state definite le seguenti probabilità di routing per la rete mostrata in precedenza:

- Probabilità di routing in uscita dall'accettazione delle richieste:
  - Uscita dal sistema: $p_{0,exit} = p_{rejected} = 0.1$.
  - Elaborazione richieste abbonati: $p_{0,1} = p_{accpeted} \cdot p_{subscriber} = 0.2916$.
  - Elaborazione richieste non abbonati: $p_{0,2} = p_{accpeted} \cdot p_{not\_subscriber} = 0.6084$.
- Le altre probabilità di routing sono pari a 1:
  - $p_{1,3} = 1$.
  - $p_{2,3} = 1$.

4.3 Costi di Gestione

Ad ogni servente è stato associato un costo di gestione mensile:

- Costo per l'accettazione delle richieste: $C_0 = 110$€.
- Costo per l'elaborazione delle richieste da parte degli utenti abbonati: $C_1 = 670$€.
- Costo per l'elaborazione delle richieste da parte degli utenti non abbonati: $C_2 = 340$€.
- Costo per la validazione dei risultati prodotti: $C_3 = 200$€.

Il costo totale viene calcolato su ogni servente per il tempo in cui questo è attivo, anche se non viene utilizzato durante tale periodo.

5 Modello Computazionale

Per simulare l'esecuzione del servizio è stato utilizzato l'approccio di tipo **Next-Event Simulation**, in cui l'avanzamento del tempo dipende dal processamento dell'evento successivo. Il modello computazionale è stato sviluppato in linguaggio `c` e il codice della simulazione è stato implementato in vari file sorgente:

- `main.c`: contiene il codice per avviare il programma.
- `config.c`: contiene una funzione per impostare la configurazione dei server della simulazione.
- `logic.c`: contiene il codice generale per eseguire una simulazione.
- `finite_horizon.c`: contiene il codice per eseguire una simulazione ad orizzonte finito.
- `infinite_horizon.c`: contiene il codice per eseguire una simulazione ad orizzonte infinito.
- `utils.c`: contiene diverse funzioni ausiliare utilizzate durante la simulazione.

Sono stati definiti, inoltre, vari file header:

- `config.h`: contiene la configurazione del sistema.
- `var.h`: contiene le variabili globali utilizzate nella simulazione.
- `structure.h`: contiene le strutture utilizzate per la simulazione.

5.1 Strutture Dati

In questa sezione vengono descritte le principali strutture dati utilizzate nel programma di simulazione:

- Struttura `network_struct` che contiene le informazione sulla rete.
- Struttura `node` che contiene le informazioni sul singolo nodo del sistema:
  - I job in coda nel nodo sono stati implementati tramite una lista collegata in cui ogni job punta a quello successivo.
  - La lista collegata mantiene tutti i job attualmente presenti nel nodo (nei serventi e nella coda).
- Struttura `server` che rappresenta un servente di un nodo.
- Struttura `clock_struct` che mantiene il tempo di simulazione e contiene tre campi:
  - `current`: indica il clock attuale di simulazione.
  - `arrival`: indica il clock del prossimo arrivo da processare.
  - `next`: indica il clock del prossimo evento da processare.
- Struttura `completions_list_struct` che contiene tutti gli eventi di completamento. Questa struttura supporta due operazioni:
  - `add_to_completions_list()`: inserisce un completamento nella lista.
  - `remove_from_completions_list()`: elimina un completamento dalla lista.

5.2 Gestione degli Eventi

Siccome gli eventi di completamento ed arrivo vengono gestiti in modo differente, è sempre necessario verificare la tipologia del prossimo evento:

1. Bisogna ottenere il prossimo completamento accedendo alla `completions_list`.
2. Il clock avanza impostando `clock.next` al minimo tra `clock.arrival` e prossimo completamento e `clock.current` a `clock.next`.
3. Se `clock.current` è uguale a `clock.arrival` bisogna gestire un arrivo, altrimenti un completamento.

5.2.1 Arrivo

Il primo tempo di arrivo viene generato chiamando la funzione `generate_arrival_time()` che sfrutta il metodo `Exponential()` di `rvgs.c`. Ogni arrivo dall'esterno viene gestito dal nodo per l'accettazione delle richieste. Bisogna verificare se esiste un servente libero in questo nodo tramite `find_free_server()` e in questo caso:

1. Viene generato un tempo di servizio esponenziale per il completamento tramite la funzione `generate_service_time()`.
2. Viene registrato il job nel servente, inserendolo nella lista collegata dei job tramite la funzione `add_job_to_queue()`.
3. Viene inserito il completamento nella `completions_list` tramite la funzione apposita.
4. Viene impostato lo stato del server come `BUSY`.

Se invece non c'è nessun server libero, il job viene inserito nella coda del nodo.

5.2.2 Completamento

Dopo aver processato un completamento, vengono rimossi il job dal server del nodo ,tramite la funzione `remove_job_from_queue()`, e il completamento dalla `completions_list`.

Il completamento di un job equivale ad un arrivo nel nodo successivo. Tramite la funzione `find_destination_node()` è possibile trovare il nodo di destinazione in base al nodo corrente secondo le diverse probabilità di routing:

- Genera un valore casuale tra 0 e 100 tramite la funzione `Uniform()`.
- Confronta questo valore con le diverse probabilità per individuare il nodo di destinazione.

5.2.3 Cambio di Fascia Oraria

Quando cambia la fascia oraria, il flusso di arrivi in ingresso varia, quindi è necessario far fronte a questi cambiamenti attraverso la riconfigurazione della rete. A livello implementativo viene utilizza la funzione `update_network()`:

- Confronta la configurazione attuale con quella della nuova fascia oraria.
- Attiva o disattiva i server necessari per ogni nodo della rete.

Se un server sta processando un job quando viene scelto per essere disattivato si attende che termini l'esecuzione attuale:

- Il server viene taggato come da disattivare al prossimo completamento attraverso il flag apposito (`stop_after_completion`).
- Questo meccanismo permette di non buttare il lavoro svolto dal server.
- Il tempo in eccesso in cui il server è attivo viene comunque conteggiato nel calcolo dei costi.

5.3 Simulazione ad Orizzonte Infinito

Nella simulazione ad orizzonte infinito il sistema viene simulato per un tempo infinito per produrre delle statistiche relative allo stato stazionario del sistema. Eseguire una simulazione di durata maggiore rispetto a quella della fascia oraria permette di ridurre la distorsione dello stato iniziale. In questo tipo di simulazione si assume che il sistema sia statico, quindi il tasso di arrivo resta costante.

Per ricavare la media campionaria del tempo di risposta è stato utilizzato il metodo delle **Batch Means**: la simulazione è stata suddivisa in `k` batch di dimensione `b`. Da ogni batch è possibile ricavare le statistiche di interesse. All'inizio di un nuovo batch, le statistiche vengono azzerate ma viene mantenuto lo stato del sistema (ovvero tutti i job del precedente batch). In questo modo viene generato un campione di `k` batch indipendenti utile per valutare la media campionaria.

Le dimensioni di `b` e `k` influiscono sulla qualità del campione: un `b` grande permette di avere un campione con una bassa autocorrelazione, mentre un `k` grande permette di avere delle stime migliori in termini di intervallo di confidenza.

Sono stati utilizzati `k` pari a **128** batch, e `b` pari a **1024**, quindi, per ogni simulazione vengono processati `b·k` job (ovvero 131072 job).
Gli obiettivi principali della simulazione ad orizzonte infinito sono stati i seguenti:

- Ricerca della configurazione ottimale che permette, per ogni fascia oraria, di rispettare i due requisiti di QoS e allo stesso tempo minimizzare i costi.
- Analisi dei tempi di risposta nello stato stazionario del sistema.
- Analisi della probabilità di perdita per il nodo di validazione della risposta.

5.4 Simulazione ad Orizzonte Finito

Nella simulazione ad orizzonte finito il sistema viene simulato per un tempo finito (in questo caso 24 ore) per produrre le statistiche transienti del sistema. Questo tipo di simulazione tiene in considerazione la variazione del tasso di arrivo e del numero di serventi attivi nelle varie fasce orarie. Il sistema si deve trovare nello stato inattivo sia all'inizio che alla fine della simulazione.

Per produrre delle statistiche ad orizzonte finito, è necessario ripetere più volte la simulazione per produrre un ensemble: ogni replica rappresenta un punto del campione che permette di analizzare le statistiche prodotte. In questo progetto la simulazione ha un ensemble di dimensione 128 (vengono eseguite 128 repliche della simulazione).

La media campionaria e l'intervallo di confidenza al 95% si possono calcolare attraverso il programma `estimate.c`, che utilizza la distribuzione di *Student-t*.

Per la prima replica il seme RNG viene impostato tramite `PlantSeeds()` fuori dal ciclo di replicazione. Per quelle successive viene usato, come stato iniziale di ogni stream RNG, lo stato finale degli stessi stream per la replica precedente. In questo modo si evitano possibili sovrapposizioni degli eventi generati dalle singole repliche.

La simulazione ad orizzonte finito termina al raggiungimento, da parte del clock, di un certo valore. In questo caso deve raggiungere le 24 ore (86400 secondi).

Per analizzare le statistiche ottenute nel continuo viene eseguita ogni 5 minuti, per ogni ripetizione, una misurazione del tempo di risposta:

- Ad intervalli di 5 minuti si ottengono 128 misurazioni del tempo di risposta.
- Calcolando la media di queste misurazioni, si ottiene il valore del tempo di risposta per ogni specifico orario.

In questo caso, la simulazione ad orizzonte finito ha come obiettivi principali:

- Analizzare il comportamento del sistema al cambio di fascia oraria, verificando che le configurazioni individuate nella simulazione ad orizzonte infinito siano valide effettivamente anche nel sistema reale.
- Confrontare diverse configurazioni per valutare il comportamento del sistema nel tempo e, in particolare, al cambio di fascia oraria.
- Analizzare i costi reali nell'arco di una giornata.

6 Verifica

In questa sezione verifichiamo che il modello computazionale è effettivamente conforme al modello delle specifiche, ovvero che l'implementazione del modello computazionale è effettivamente corretta. Per la verifica bisogna considerare le seguenti quattro condizioni:

1. Il numero di arrivi in un nodo è sempre uguale al numero di job in coda più il numero di job completati.

2. Il tempo di risposta per un server è sempre uguale alla somma del tempo di attesa e del tempo di servizio.

3. Il numero di job in ingresso è conforme a quello indicato nel modello delle specifiche.

4. Dato un certo numero di job in ingresso nel sistema, le percentuali di job in ingresso in ogni nodo sono conformi al tasso di arrivo e alle probabilità di routing specificate.

Per valutare queste condizioni sono state eseguite varie simulazioni ad orizzonte finito e infinito. Consideriamo i risultati prodotti da una di queste simulazioni:

```
Node REQUEST_ACCEPTANCE
Arrivals ................. = 270533
Completions .............. = 242531
Job dropped .............. = 28002
Average wait ............. = 1.517296
Average delay ............ = 0.518537
Average service time ..... = 0.998759
```

```
Node NON_SUBSCRIBER_REQUEST_PROCESSING
Arrivals ................. = 163917
Completions .............. = 163917
Average wait ............. = 17.622092
Average delay ............ = 2.651941
Average service time ..... = 14.970151
```

```
Node SUBSCRIBER_REQUEST_PROCESSING
Arrivals ................. = 78614
Completions .............. = 78614
Average wait ............. = 5.897204
Average delay ............ = 0.904947
Average service time ..... = 4.992257
```

```
Node RESPONSE_VALIDATION
Arrivals ................. = 242531
Completions .............. = 194598
Job lost ................. = 47933
Average wait 1 ........... = 2.991590
Average wait 2 ........... = 2.400342
```

Per la prima condizione:

- Nodo accettazione richieste: 270533 = 242531 + 28002.
- Nodo elaborazione richieste abbonati: 78614 arrivi e 78614 completamenti.
- Nodo elaborazione richieste non abbonati: 163917 arrivi e 163917 completamenti.
- Nodo validazione riposta: 242531 = 194598 + 47933.

Per la seconda condizione:

- Nodo accettazione richieste: 1.517296 = 0.518537 + 0.998759.
- Nodo elaborazione richieste abbonati: 5.897204 = 0.904947 + 4.992257.
- Nodo elaborazione richieste non abbonati: 17.622092 = 2.651941 + 14.970151.
- Nodo validazione risposta: in questo caso bisogna verificare che la media tra il tempo di risposta per i soli job che entrano in servizio e quelli persi corrisponda al valore `Average wait 2`, che indica il tempo di risposta medio per un qualsiasi job (che entra in servizio o che salta la validazione nel nodo finale):

$$\frac{(242531 - 47933) \cdot 2.991590 + 47933 \cdot 0}{242531} = 2.400342$$

Per la terza condizione:

- Nel nodo di accettazione il numero di arrivi è pari a 270533, che è di poco superiore al numero di utenti medi al giorno (270000).
- Questa discrepanza è data dal fatto che gli arrivi vengono generati secondo un processo stocastico.

- Calcolando la media degli arrivi su tutte le 128 ripetizioni della simulazione si ottiene un valore di arrivi molto più vicino al modello delle specifiche:

```
Average arrivals for 128 repetitions:  269994
```

Per la quarta condizione:

- Per le richieste non valide ($p_{0,exit} = 0.1$): $270533 \cdot 0.1 = 27053.3 \approx 28002$.
- Per le richieste eseguite da utenti abbonati al servizio ($p_{0,1} = 0.2916$): $270533 \cdot 0.2916 = 78887.4228 \approx 78614$.
- Per le richieste eseguite da utenti non abbonati al servizio ($p_{0,2} = 0.6084$): $270533 \cdot 0.6084 = 164592.2772 \approx 163917$.
- Per le richieste già processate basta vedere che il numero di richieste valide è pari al numero di job in ingresso al nodo di validazione (242531).

7 Validazione

In questa sezione verifichiamo che il modello computazionale è coerente con il sistema reale. Non avendo a disposizione dei dati reali, possiamo effettuare una validazione rispetto al modello analitico, verificando che i risultati ottenuti rispettino le leggi teoriche.

Per ottimizzare la fase di validazione è stato utilizzato il foglio *Excel* citato in precedenza che contiene anche diverse leggi teoriche calcolate automaticamente in base ai parametri di input che vengono specificati.

La validazione del sistema è stata effettuata ad orizzonte finito utilizzando la seguente configurazione di server casuale:

Configurazione	**Nodo 0**	**Nodo 1**	**Nodo 2**	**Nodo 3**
02:00 - 09:00	3	4	17	5
09:00 - 21:00	5	8	40	10
21:00 - 02:00	4	6	30	8

Nella fase di validazione bisogna confrontare i risultati della simulazione rispetto al modello analitico, controllando le seguenti leggi:

- $E(TQ_k) = P_Q \cdot \dfrac{\rho}{\lambda(1-\rho)}$ per $k \in [0, 2]$
  - $P_Q = \dfrac{(m\rho)^m}{m!(1-\rho)} \cdot P(0)$
  - $P(0) = \dfrac{1}{\sum_0^{m-1} \dfrac{(m\rho)^i}{i!} + \dfrac{(m\rho)^m}{m!(1-\rho)}}$
- $E(T_{s,k}) = E(TQ_k) + E(S_i)$ per $k \in [0, 2]$
- $P_{loss} = \pi_m = \dfrac{\dfrac{1}{m} \cdot (\dfrac{\lambda}{\mu})^m}{\sum_0^m (\dfrac{\lambda}{\mu})^j \cdot \dfrac{1}{j!}}$
- $E(T_{s,tot}) = \sum_0^3 \nu_k \cdot E(T_{s,k})$

Per il nodo di validazione della risposta (coda **M/M/k/k**) abbiamo che:

- Il tempo di coda è sempre nullo, in quanto non è presente una coda:

$$E(TQ_4) = 0$$

- Nel modello analizzato i job che non eseguono la validazione nel nodo sono comunque completati con tempo di servizio nullo:

$$E(T_{s,4}) = P_{loss} \cdot 0 + (1 - P_{loss}) \cdot E(S_4)$$

Attraverso la tabella seguente è possibile vedere che il modello simulativo rispetta le leggi del modello analitico in base a un intervallo di confidenza del 95%:

Fascia Oraria 1: 02:00 - 09:00		
Statistica	Risultato Analitico	Risultato Sperimentale ($\alpha = 0.05$)
$E(TQ_0)$	0,198533364	0.197530742 ± 0.001569148
$E(TQ_1)$	0,812423498	0.809154188 ± 0.014722222
$E(TQ_2)$	2,944658307	2.910661547 ± 0.080169400
P_{loss}	0,229083452	0.228590430 ± 0.000541805
$E(T_{s,tot})$	15,89244448	15.904031641 ± 0.064375674

Fascia Oraria 2: 09:00 - 21:00		
Statistica	Risultato Analitico	Risultato Sperimentale ($\alpha = 0.05$)
$E(TQ_0)$	0,617026933	0.598512453 ± 0.023035127
$E(TQ_1)$	0,816790556	0.818266891 ± 0.008555449
$E(TQ_2)$	2,752637016	2.793019289 ± 0.058760378
P_{loss}	0,258211251	0.252261664 ± 0.008260110
$E(T_{s,tot})$	16,11673704	16.059413070 ± 0.066491645

Fascia Oraria 1: 21:00 - 02:00		
Statistica	Risultato Analitico	Risultato Sperimentale ($\alpha = 0.05$)
$E(TQ_0)$	0,509433962	0.532456359 ± 0.035801474
$E(TQ_1)$	1,181990277	0.987487773 ± 0.218474836
$E(TQ_2)$	3,000116447	2.840082438 ± 0.171391995
P_{loss}	0,241103226	0.250105211 ± 0.009259846
$E(T_{s,tot})$	16,31239446	16.128680555 ± 0.434186936

8 Progettazione delle Simulazioni

Durante la fase di simulazione, sono state provate diverse configurazioni di server per il sistema nelle tre fasce orarie. La configurazione dei server viene gestita da una struttura apposita (`network_configuration`) che contiene, per ogni nodo, il numero di server attivi in ogni fascia oraria.

Sono state eseguite diverse simulazioni ad orizzonte finito e infinito a partire da queste configurazioni. I risultati delle varie esecuzioni sono stati salvati su dei file `csv`, analizzabili tramite i programmi `estimate` e `uvs` per ottenere i valori di media, deviazione standard ed intervallo di confidenza.

9 Analisi dei Risultati

9.1 Analisi per Simulazione ad Orizzonte Infinito

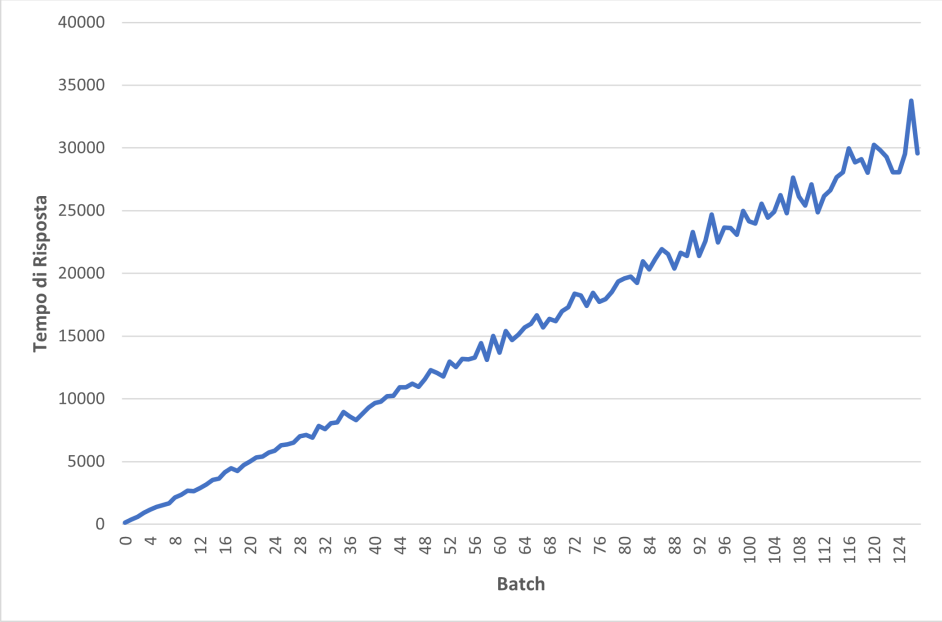
Durante l'esecuzione delle simulazioni ad orizzonte infinito sono state cercate le configurazioni ottimali, per ogni fascia oraria, in grado di rispettare i requisiti di QoS specificati in precedenza e minimizzare i costi.

9.1.1 Fascia Oraria 1: 02:00 - 09:00

Consideriamo la seguente configurazione:

{1,4,17,5}

Viene prodotto il seguente tempo di risposta:



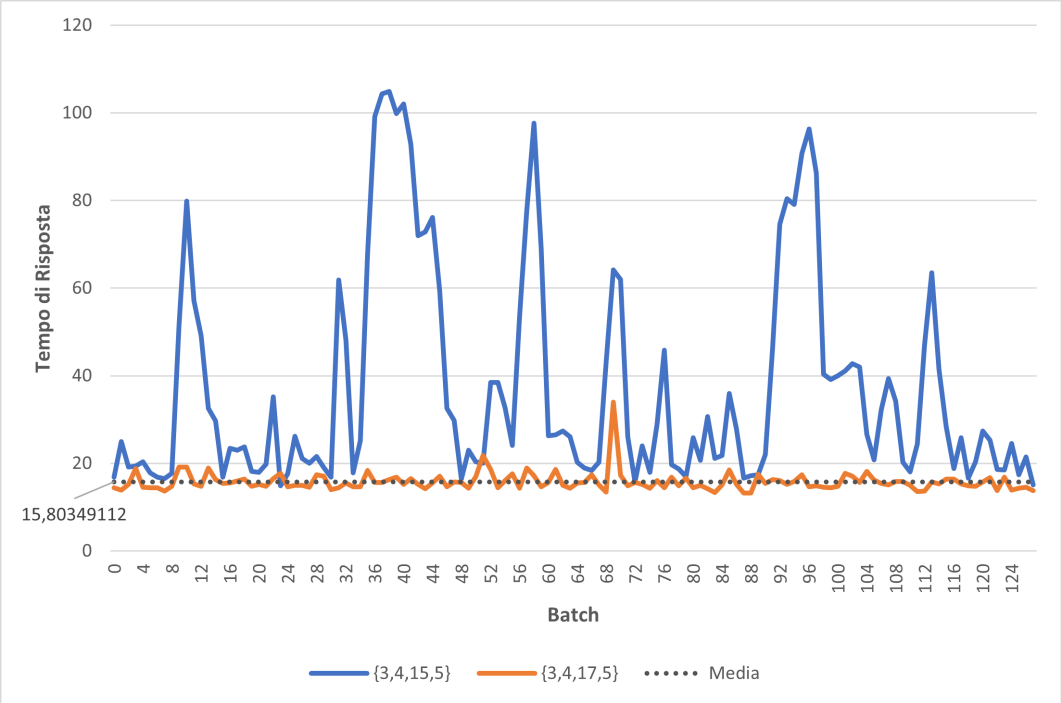
Possiamo notare che con questa configurazione non viene mai raggiunto lo stato stazionario in quanto il sistema non è stabile e il tempo di risposta aumenta all'infinito. Infatti, il nodo per l'accettazione delle richieste ha un'utilizzazione pari a 1, quindi la coda di questo nodo cresce all'infinito in quanto non riesce a processare tutti i job in ingresso.

```
Utilization REQUEST_ACCEPTANCE: ..................... 1.000000
Utilization SUBSCRIBER_REQUEST_PROCESSING: ........... 0.362399
Utilization NON_SUBSCRIBER_REQUEST_PROCESSING: ....... 0.532918
Utilization RESPONSE_VALIDATION: ..................... 0.490078
-----------------------------------------------------------------
Loss percentage: ..................................... 0.084614
-----------------------------------------------------------------
Configuration cost: ................................. 379.69€
```

Confrontiamo due configurazioni simili:

{3,4,<u>15</u>,5} e **{3,4,<u>17</u>,5}**

Vengono prodotti i seguenti tempi di risposta:



Possiamo osservare che per la configurazione 3,4,15,5 il sistema produce un tempo di risposta altamente variabile che supera, a tratti, abbondantemente il requisito di QoS (25 secondi). Questo dipende principalmente dall'alta utilizzazione molto alta del blocco di elaborazione delle richieste degli utenti non abbonati, per cui il traffico all'interno del nodo viene smaltito più lentamente.

```
Utilization REQUEST_ACCEPTANCE: ..................... 0.533570
Utilization SUBSCRIBER_REQUEST_PROCESSING: ........... 0.584174
Utilization NON_SUBSCRIBER_REQUEST_PROCESSING: ....... 0.971923
Utilization RESPONSE_VALIDATION: ..................... 0.668697
-------------------------------------------------------------
Loss percentage: ..................................... 0.227698
-------------------------------------------------------------
Configuration cost: ................................. 361.09€
```

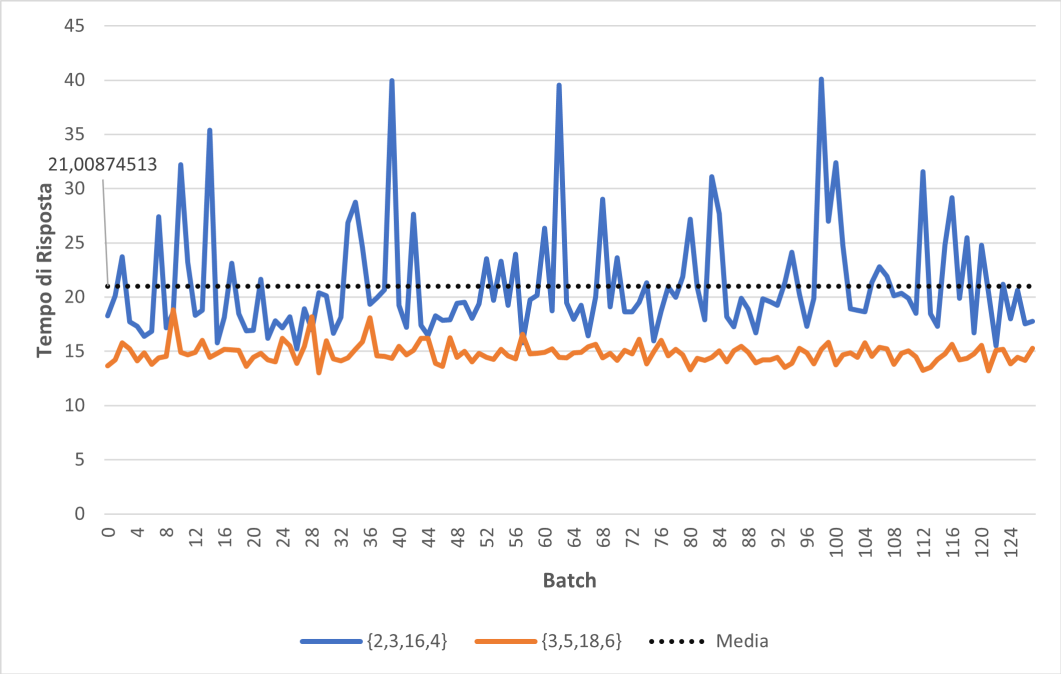
Possiamo migliorare la configurazione aggiungendo solo due server nel nodo in modo tale che l'utilizzazione diminuisca. Il costo della configurazione cresce di poco e il tempo di risposta rimane molto più stabile con un valore medio pari a 15.80 ± 0.38 che rispetta il requisito. La percentuale di risposte che non vengono validate rimane invariata tra le due configurazione, ma in entrambi i casi viene rispettato il vincolo del 35%.

```
Utilization REQUEST_ACCEPTANCE: ..................... 0.533873
Utilization SUBSCRIBER_REQUEST_PROCESSING: ........... 0.587916
Utilization NON_SUBSCRIBER_REQUEST_PROCESSING: ....... 0.858192
Utilization RESPONSE_VALIDATION: ..................... 0.668403
-------------------------------------------------------------
Loss percentage: ..................................... 0.228094
-------------------------------------------------------------
Configuration cost: ................................. 388.19€
```

Consideriamo la configurazione {3,5,18,6} che rispetta entrambi i requisiti di QoS:

- $E(T_{s,tot})$: $14.80 \pm 0.16 < 25$ secondi.
- P_{loss}: $0.14 < 0.35$.

Siccome le utilizzazioni sono abbastanza inferiori a 1, possiamo migliorare la configurazione rimuovendo alcuni server. Passiamo, dunque, alla configurazione {2,3,16,4} e confrontiamo i risultati ottenuti:



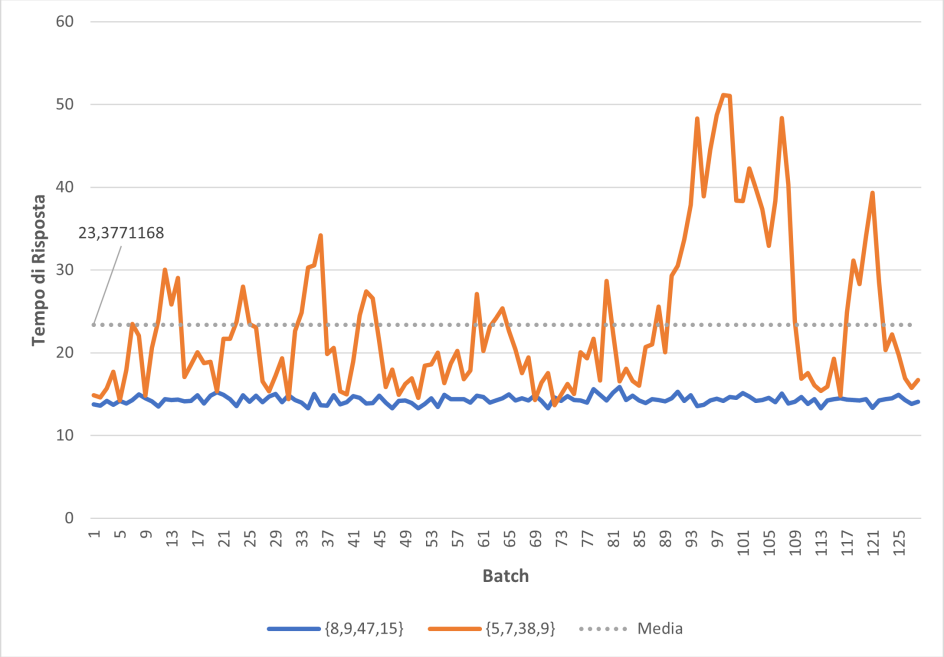
Possiamo vedere come la seconda configurazione varia molto di più rispetto alla prima in quanto le utilizzazione dei vari nodi sono maggiori. Nonostante ciò, il vincolo sul tempo di risposta viene rispettato (21.01 ± 0.86). Anche il vincolo sulla percentuale di risposte da validare viene soddisfatto ($0.34 < 0.35$). Possiamo vedere che i costi diminuiscono di molto: si scende da 435.98€ a 335.75€.

```
Utilization REQUEST_ACCEPTANCE: ..................... 0.802073
Utilization SUBSCRIBER_REQUEST_PROCESSING: ........... 0.773236
Utilization NON_SUBSCRIBER_REQUEST_PROCESSING: ....... 0.912027
Utilization RESPONSE_VALIDATION: ..................... 0.710046
-----------------------------------------------------------------
Loss percentage: ..................................... 0.342806
-----------------------------------------------------------------
Configuration cost: ................................. 335.75€
```

Rimuovendo ulteriormente dei server da questa configurazione, è possibile vedere che non vengono più rispettati i requisiti di QoS. Quindi, la configurazione **{2,3,16,4}** rappresenta quella ottimale per la prima fascia oraria.

9.1.2 Fascia Oraria 2: 09:00 - 21:00

Consideriamo le seguenti configurazioni {8,9,47,15} e {5,7,38,9}:



Possiamo osservare che, per la prima configurazione, le utilizzazioni sono molto basse e vengono rispettati ampiamente i requisiti di QoS:

- $E(T_{s,tot})$: $14.31 \pm 0.09 < 25$ secondi.
- P_{loss}: $0.06 < 0.35$.

```
Utilization REQUEST_ACCEPTANCE: ..................... 0.507138
Utilization SUBSCRIBER_REQUEST_PROCESSING: ........... 0.658804
Utilization NON_SUBSCRIBER_REQUEST_PROCESSING: ....... 0.788050
Utilization RESPONSE_VALIDATION: ..................... 0.685562
-------------------------------------------------------------
Loss percentage: ..................................... 0.059241
-------------------------------------------------------------
Configuration cost: ................................. 405.13€
```

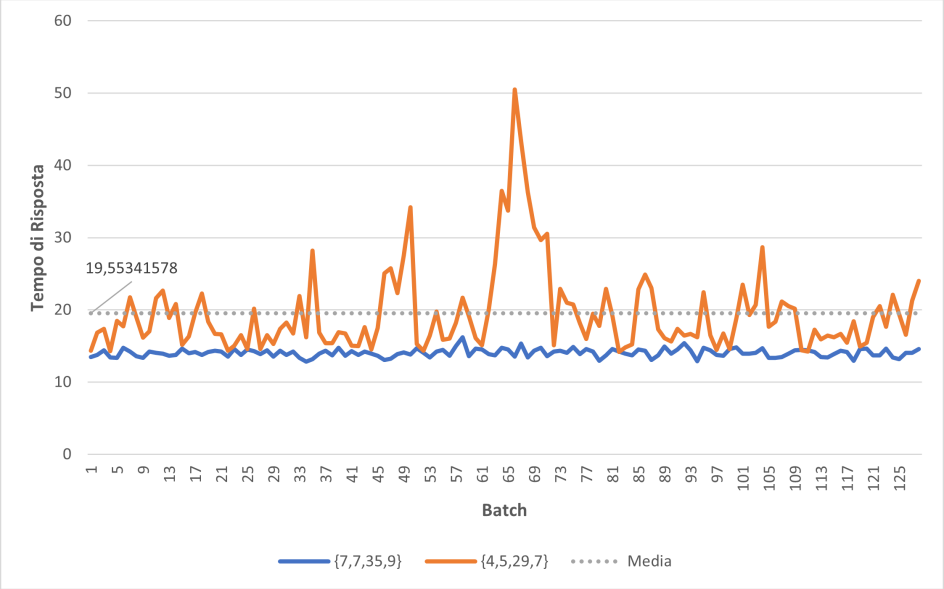
Per la seconda configurazione le utilizzazioni sono molto più alte e, di conseguenza, aumenta il tempo di risposta medio che rispetta comunque il primo requisito (23.38 ± 1.56). Anche la probabilità che una risposta non venga validata aumenta ($0.32 < 0.35$). Questa configurazione permette di ridurre notevolmente i costi: da 405.13€ a 313.15€.

```
Utilization REQUEST_ACCEPTANCE: ..................... 0.807017
Utilization SUBSCRIBER_REQUEST_PROCESSING: ........... 0.847265
Utilization NON_SUBSCRIBER_REQUEST_PROCESSING: ....... 0.973103
Utilization RESPONSE_VALIDATION: ..................... 0.826836
-------------------------------------------------------------
Loss percentage: ..................................... 0.318032
-------------------------------------------------------------
Configuration cost: ................................. 313.15€
```

Rimuovendo ulteriormente dei server da quest'ultima configurazione, è possibile vedere che non vengono più rispettati i requisiti di QoS. Quindi, la configurazione **{5,7,38,9}** rappresenta quella ottimale per la seconda fascia oraria.

9.1.3 Fascia Oraria 3: 21:00 - 02:00

Consideriamo le seguenti configurazioni {7,7,35,9} e {4,5,29,7}:



Possiamo osservare che, per la prima configurazione, le utilizzazioni sono molto basse e vengono rispettati ampiamente i requisiti di QoS:

- $E(T_{s,tot})$: $14.04 \pm 0.10 < 25$ secondi.
- P_{loss}: $0.18 < 0.35$.

```
Utilization REQUEST_ACCEPTANCE: ..................... 0.426244
Utilization SUBSCRIBER_REQUEST_PROCESSING: ........... 0.631359
Utilization NON_SUBSCRIBER_REQUEST_PROCESSING: ....... 0.777386
Utilization RESPONSE_VALIDATION: ..................... 0.736954
-----------------------------------------------------------------
Loss percentage: ..................................... 0.177242
-----------------------------------------------------------------
Configuration cost: ................................. 407.07€
```

Per la seconda configurazione le utilizzazioni sono molto più alte e, di conseguenza, aumenta il tempo di risposta medio che rispetta comunque il primo requisito (19.55 ± 1.02). Anche la probabilità che una risposta non venga validata aumenta ($0.31 < 0.35$). Questa configurazione permette di ridurre notevolmente i costi: da 407.07€ a 320.05€.

```
Utilization REQUEST_ACCEPTANCE: ..................... 0.746926
Utilization SUBSCRIBER_REQUEST_PROCESSING: ........... 0.871439
Utilization NON_SUBSCRIBER_REQUEST_PROCESSING: ....... 0.941471
Utilization RESPONSE_VALIDATION: ..................... 0.792273
-----------------------------------------------------------------
Loss percentage: ..................................... 0.313735
-----------------------------------------------------------------
Configuration cost: ................................. 320.05€
```

Rimuovendo ulteriormente dei server da quest'ultima configurazione, è possibile vedere che non vengono più rispettati i requisiti di QoS. Quindi, la configurazione **{4,5,29,7}** rappresenta quella ottimale per la terza fascia oraria.

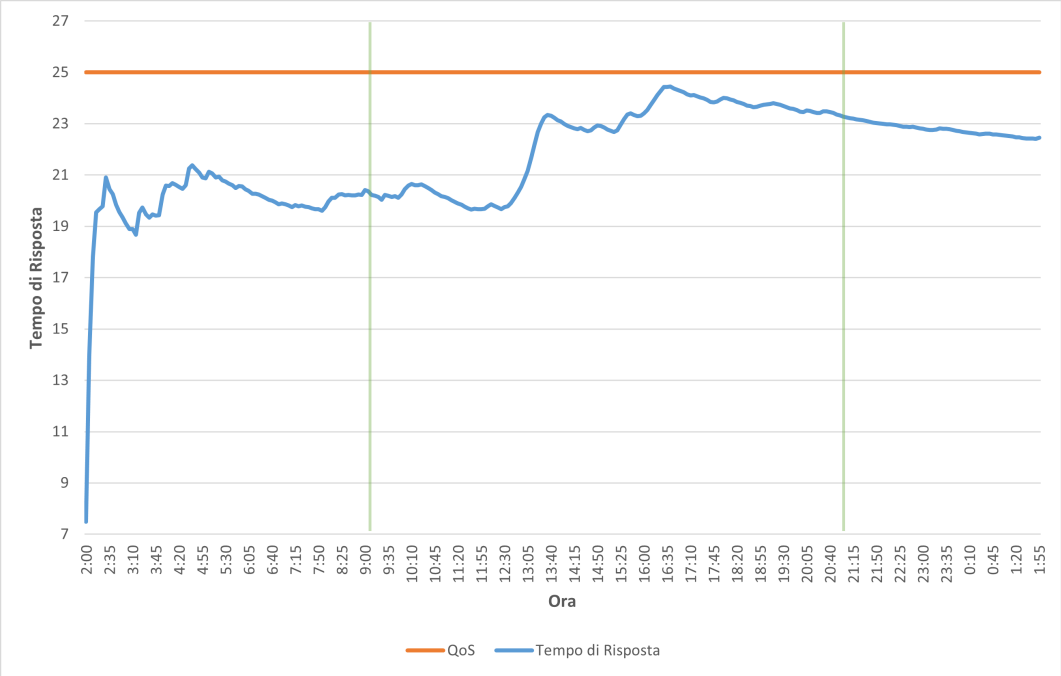
9.2 Analisi per Simulazione ad Orizzonte Finito

Nella simulazione ad orizzonte finito sono state analizzate diverse configurazioni per valutare il comportamento transiente del sistema al cambio di fascia oraria. È stata individuata la configurazione ottima che permette di minimizzare i costi rispettando i requisiti di QoS:

`{2,3,16,4}, {5,7,38,9}, {4,5,29,7}`

9.2.1 Configurazione Ottima - {2,3,16,4}, {5,7,38,9}, {4,5,29,7}

Possiamo analizzare il comportamento nel transiente della configurazione ottima:



Possiamo vedere dal grafico che, in tutte le fasce orarie, il tempo di risposta medio rispetta sempre il requisito di 25 secondi.

- Nella prima fascia oraria il tempo di risposta parte molto basso per poi stabilizzarsi al valore medio verso la fine (circa alle 8:30). Questo comportamento è atteso in quanto il sistema inizialmente è nello stato vuoto, quindi per i primi arrivi si hanno dei tempi di risposta molto bassi.
- Quando si passa nella seconda fascia oraria, il tempo di risposta rimane stabile, ma sale nella seconda parte a causa del notevole aumento di traffico che, riempendo le code, causa un aumento del tempo di attesa. In questo orario il tempo di risposta tende a non avere un valore stabile.
- Nella terza fascia oraria, invece, il tempo di risposta tende a decrescere linearmente, in quanto la quantità di traffico non decresce drasticamente (si passa da circa 4 arrivi al secondo a 3 arrivi al secondo). Il tempo di risposta si stabilizza solo alla fine della fascia oraria.

I tempi di risposta medi per le varie fasce orarie sono i seguenti:

- Fascia 1: $\mathrm{E}(\mathrm{T}_{\mathrm{s,tot}}) = 20.86 \pm 0.14$.
- Fascia 2: $\mathrm{E}(\mathrm{T}_{\mathrm{s,tot}}) = 22.27 \pm 0.28$.
- Fascia 3: $\mathrm{E}(\mathrm{T}_{\mathrm{s,tot}}) = 21.75 \pm 0.23$.

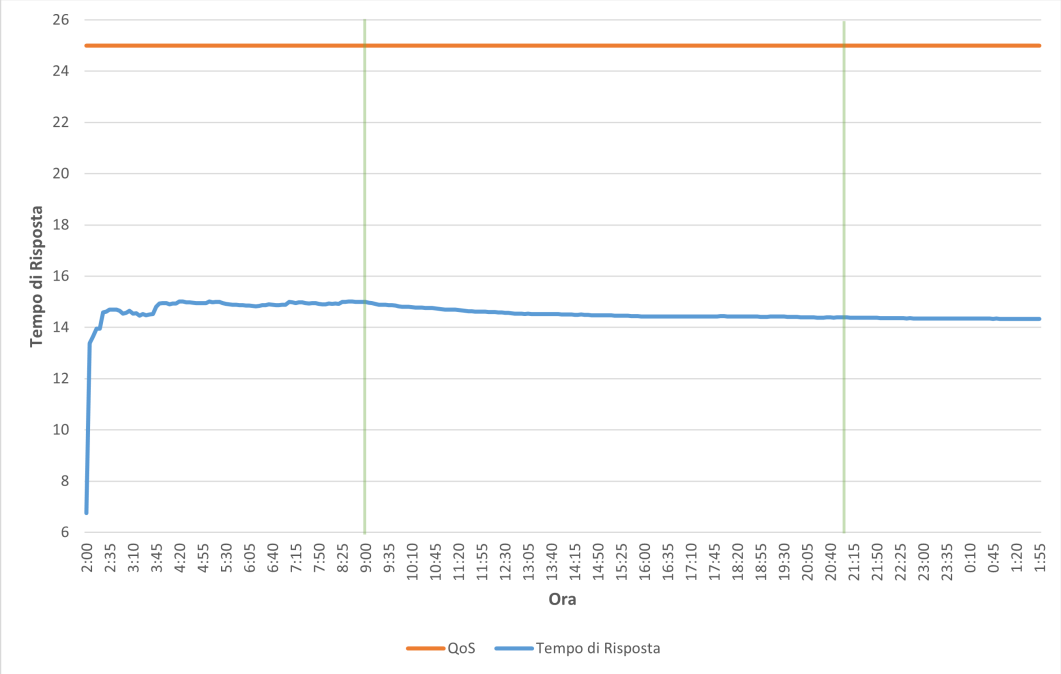
Anche la probabilità che una risposta fornita dal servizio non venga validata rispetta il requisito di 0.35 nelle tre fasce orarie:

- Fascia 1: $P_{loss} = 0.34$.
- Fascia 2: $P_{loss} = 0.32$.
- Fascia 3: $P_{loss} = 0.32$.

Il costo totale di questa configurazione è pari a 656.27€.

9.2.2 Configurazione Over Provisioning - {3,5,18,6}, {8,9,47,15}, {7,7,35,9}

Consideriamo una configurazione in cui il numero di server è sovradimensionato rispetto alla configurazione ottima, per cui il tempo di risposta medio è minore:



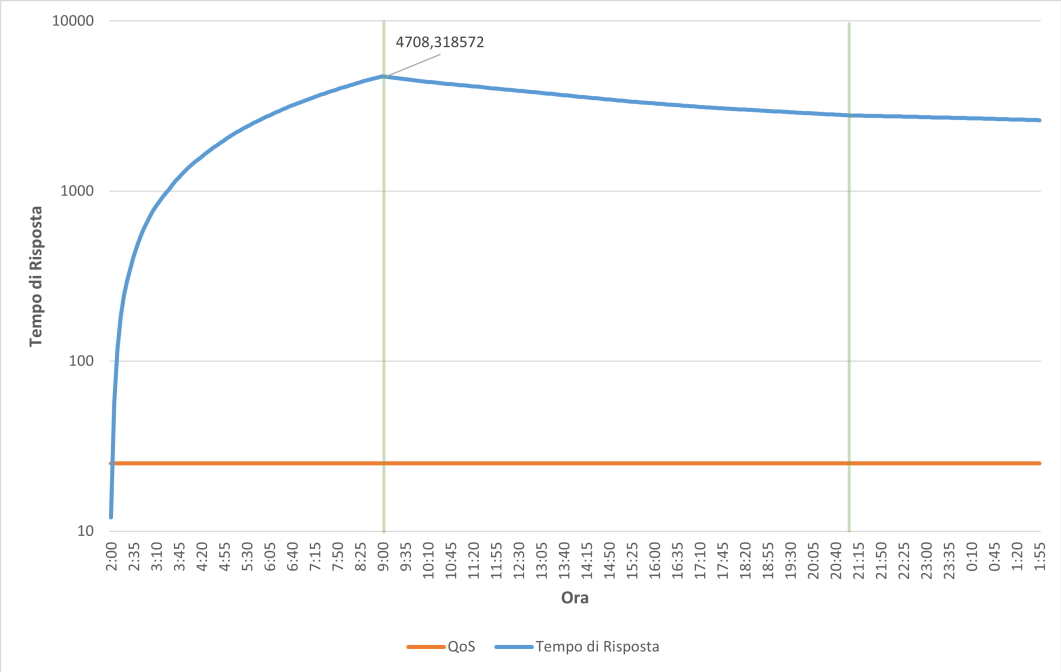
In questa configurazione, al cambio di fascia oraria, non si verificano cambiamenti drastici al tempo di risposta ma rimane molto stabile in quanto le code tendono a non riempirsi come nella configurazione precedente. In questo caso vengono rispettati ampiamente entrambi i requisiti:

- Fascia 1: $E(T_{s,tot}) = 15.02 \pm 0.03$ e $P_{loss} = 0.14$.
- Fascia 2: $E(T_{s,tot}) = 14.43 \pm 0.01$ e $P_{loss} = 0.07$.
- Fascia 3: $E(T_{s,tot}) = 14.36 \pm 0.01$ e $P_{loss} = 0.09$.

Questa configurazione risulta sicuramente più efficiente in termini di tempo di risposta e probabilità che una risposta venga validata, ma i costi sono molto più alti: 848.22€ contro i 656.27€ della configurazione ottima.

9.2.3 Configurazione Under Provisioning - {1,4,17,5}, {5,7,38,9}, {4,5,29,7}

Consideriamo una configurazione in cui, nella prima fascia oraria, il numero di server è sottodimensionato rispetto alla configurazione ottima, per cui il tempo di risposta medio è maggiore:



Analizzando il grafico è possibile vedere che nella prima fascia oraria il tempo di risposta aumenta fino a un massimo di 4708,32 secondi in quanto il sistema è instabile. Al cambio di fascia oraria vengono attivati molti server: questi consentono di smaltire più velocemente i job rimasti in coda nella prima fascia. Il tempo di risposta tende a decrescere abbastanza rapidamente ma non riesce mai a raggiungere il requisito di 25 secondi.

Possiamo misurare i seguenti tempi di risposta medi e probabilità di non validare una risposta:

- Fascia 1: $E(T_{s,tot}) = 4764.56 \pm 14.23$ e $P_{loss} = 0.09$.
- Fascia 2: $E(T_{s,tot}) = 2771.95 \pm 13.41$ e $P_{loss} = 0.31$.
- Fascia 3: $E(T_{s,tot}) = 2598.62 \pm 15.82$ e $P_{loss} = 0.31$.

Il costo di questa configurazione è pari a 669.77€ che risulta essere maggiore rispetto a quello della configurazione ottima, nonostante fornisca prestazioni molto peggiori.

10 Conclusioni

Per il sistema analizzato è stata individuata la seguente configurazione ottima:

`{2,3,16,4}, {5,7,38,9}, {4,5,29,7}`

Questa configurazione comporta un costo totale di 656.27€ con le seguenti statistiche:

- Fascia 1: $E(T_{s,tot}) = 20.86 \pm 0.14$ e $P_{loss} = 0.34$.
- Fascia 2: $E(T_{s,tot}) = 22.27 \pm 0.28$ e $P_{loss} = 0.32$.
- Fascia 3: $E(T_{s,tot}) = 21.75 \pm 0.23$ e $P_{loss} = 0.32$.

Quindi vengono rispettati entrambi i requisiti di QoS:

- $E(T_{s,tot}) < 25$ secondi.
- $P_{loss} < 0.35$.