Architettura a Microservizi

Progettazione, simulazione e valutazione delle prestazioni

Jacopo Fabi 0293870
Davide Bianchi 0293906

AA 2022/2023

Indice

1 Introduzione

Il continuo sviluppo in ambito tecnologico, con il passare del tempo, ha generato soluzioni innovative per le aziende, molte delle quali si sono adeguate così da organizzare e gestire al meglio la loro infrastruttura IT.

Il caso di studio in esame prende in considerazione un'architettura a micro-servizi di proprietà di un'azienda che offre al pubblico la possibilità di effettuare prenotazioni per viaggi verso mete turistiche.

L'azienda offre più alternative ai clienti, per questo motivo l'architettura presenta più micro-servizi, ognuno dei quali si occupa di gestire una precisa prenotazione: hotel, volo e automobile.

- Gli utenti che vogliono prenotare un biglietto aereo hanno la possibilità di prenotare un pacchetto completo aggiungendo un soggiorno in hotel e il noleggio di un'automobile per tutto il periodo di residenza.
- Gli utenti che vogliono prenotare un soggiorno in hotel hanno solamente la possibilità di aggiungere il noleggio di un'automobile per tutto il periodo di residenza.

Per completare la prenotazione del pacchetto vacanza è necessario che il pagamento vada a buon fine, per questo motivo ogni utente verrà re-direzionato, come ultimo step, verso un micro-servizio che si occupa della convalida finale: la prenotazione verrà confermata solamente se il pagamento dell'utente non presenta problemi.

2 Obiettivo

L'obiettivo dello studio è quello di minimizzare i costi di gestione dell'infrastruttura, in particolare si vuole individuare il numero ottimale di serventi per oguno dei sottosistemi presenti.

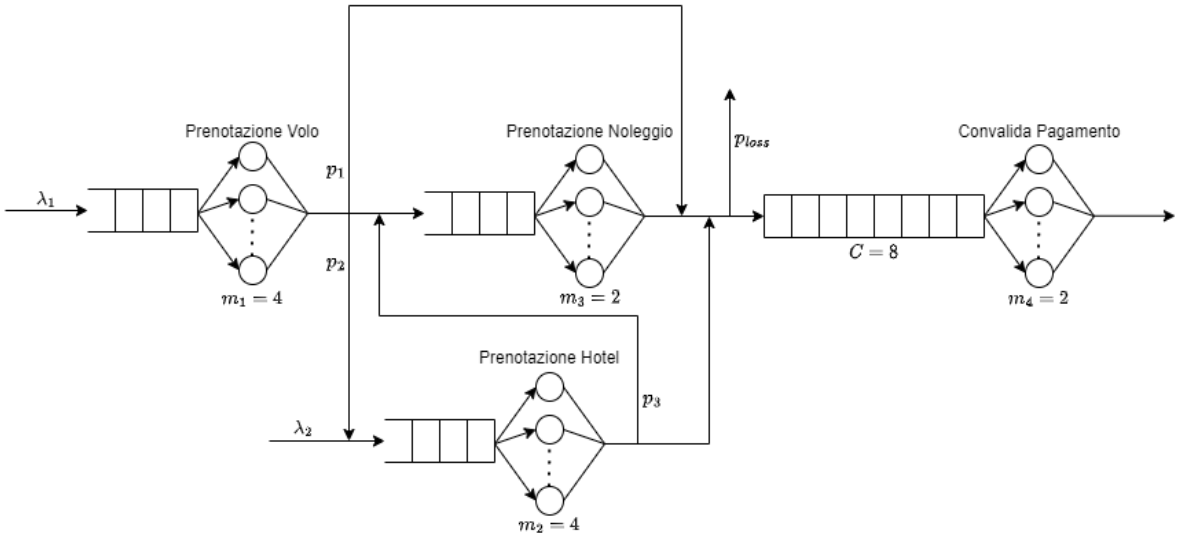
I due **QoS** che si vogliono rispettare sono:

- Mantenere il massimo tempo medio di risposta del sistema sotto i 12 secondi, senza considerare i tempi di percorrenza tra i vari micro-servizi della architettura.
- Convalidare il pagamento almeno per il 95% degli utenti che devono portare a termine una prenotazione.

A tale scopo si effettua uno studio sia dello stato stazionario che di quello transiente, variando le configurazioni dei serventi per cercare di individuare quale di queste rispetta i due vincoli fissati, ottenendo il minor costo totale (inteso come numero di serventi aggiunti).

Per lo studio in esame, in riferimento al secondo QoS, non interessa il numero di clienti per cui il pagamento è accettato o viceversa, ma l'analisi si focalizza sulla riduzione al minimo degli utenti che non riescono ad effettuare l'operazione di convalida, a prescindere dall'esito.

3 Modello Concettuale



L'architettura esistente dell'azienda è stata modellata secondo la rete in figura, ogni micro-servizio è in esecuzione su un server dedicato che presenta molteplici repliche, ciascuna eseguita su un container.
Dalla rete si possono evidenziare quattro *nodi* (o sottosistemi) differenti:

- Nodo 1: Prenotazione Volo
- Nodo 2: Prenotazione Hotel
- Nodo 3: Prenotazione Noleggio
- Nodo 4: Convalida Pagamento

Un cliente ha la possibilità di effettuare prenotazioni per pacchetti differenti in base alle sue necessità: la prenotazione del volo permette di includere nel pacchetto un soggiorno in hotel e il noleggio di un'automobile, la prenotazione dell'hotel è invece dedicata a quei clienti che non hanno la necessità di acquistare biglietti aerei ma che in caso di necessità possono noleggiare un'automobile.

I primi tre nodi sono modellati come una **M/M/m**, le code sono infinite proprio perché non si vuole limitare il numero di clienti che possono usufruire del servizio di prenotazione: all'arrivo di una richiesta su un server, si verifica se c'è una replica libera che può gestirla (in tal caso si sceglie il servente idle da più tempo), altrimenti la richiesta viene inserita in coda.

Il nodo per la convalida del pagamento è modellato come una **M/M/m/C**, quindi se tutti i serventi sono occupati la richiesta di prenotazione verrà semplicemente scartata senza neanche provare a convalidare il pagamento: la coda limitata è stata scelta al fine di garantire la stabilità del nodo, così da permettere la corretta gestione delle richieste non scartate, proprio per questo l'obiettivo dello studio è quello di dimensionare la rete in modo tale che il numero di clienti che non riescono a confermare la prenotazione sia al più del 5%.

Le **variabili di stato** considerate sono:

- Stato di un servente (IDLE/BUSY)
- Numero di richieste in uno specifico nodo

Gli **eventi** che possono verificarsi sono:

- Arrivo di una richiesta
- Servizio di una richiesta presso il servente assegnato

Gli **utenti** in ingresso al sistema sono di due tipi e questa distinzione viene utilizzata solamente all'interno dell'algoritmo migliorativo:

- Iscritti
- Visitatori

Ogni servente attivo implica un costo più elevato di gestione, per cui l'obiettivo è quello di identificare il numero minimo di serventi che sia in grado di soddisfare anche un carico di lavoro molto alto minimizzando le spese e rispettando i due QoS.

Un'architettura a micro-servizi offre al pubblico un servizio web che è disponibile 24 ore al giorno, con un numero di clienti che può variare nell'arco della giornata, ma che comunque non permette di identificare slot orari con tassi di richieste differenti:

- Essendo il servizio fruibile da ogni parte del mondo, eventuali picchi diurni sono mitigati in media dalla quasi totale inattività notturna dall'altra parte del mondo, garantendo una certa stabilità degli arrivi.
- Inoltre, un sistema distribuito utilizza tecniche come l'autoscaling che permettono al sistema di reagire a variazioni improvvise del carico adattando a run-time il numero di serventi sui diversi nodi.
- Ha poco senso quindi effettuare un dimensionamento per fasce orarie nella singola giornata, ma risulta comunque complesso simulare l'autoscaling dovendo costantemente analizzare il tasso d'arrivo nell'ultimo periodo, ad esempio negli ultimi 30 secondi, per adattare il sistema.
- Per questo motivo, lo studio mira a dimensionare l'architettura a livello della singola giornata, considerando un tasso di arrivo costante che tiene conto di tutte le considerazioni appena fatte, ipotizzando che i picchi di traffico vengono bilanciati dall'inattività di un gran numero di utenti.

Per le considerazioni fatte in precedenza, è chiaro che:

- L'analisi dello stato stazionario è quella più significativa ai fini dello studio e del dimensionamento del sistema.
- L'analisi transiente fornisce informazioni più dettagliate sul comportamento del sistema e sui tempi di convergenza.

4 Modello delle Specifiche

4.1 Dati di Input

Per la scelta dei parametri di input non è stato trovato alcun dato disponibile al pubblico sulle richieste che un servizio web riceve in un certo periodo, informazioni che avrebbero permesso di produrre una stima verosimile del tasso di arrivo medio. La configurazione iniziale del sistema è la seguente:

- m_k = numero di serventi del nodo k
  - $m_1 = 4$
  - $m_2 = 4$
  - $m_3 = 2$
  - $m_4 = 2$
- C_k = capacità della coda del nodo k
  - $C_1 = \infty$
  - $C_2 = \infty$
  - $C_3 = \infty$
  - $C_4 = 8$
- Per semplicità, si utilizza $C_4 = C$ essendo l'unico nodo con coda finita

I tempi di servizio e interarrivo sono modellati con una distribuzione **Esponenziale** che, in generale, è un'ottima scelta per rappresentare occorrenze randomiche:

- Numero di clienti al giorno: 240.000 circa
  - 1,9 arrivi medi al secondo per prenotazioni di voli
  - 0,8 arrivi medi al secondo per prenotazioni di hotel
- Tempo di prenotazione volo: 2 secondi
  - 0.5 prenotazioni al secondo
- Tempo di prenotazione hotel: 3,2 secondi
  - 0,3 prenotazioni al secondo
- Tempo di prenotazione noleggio: 2,5 secondi
  - 0,4 prenotazioni al secondo
- Tempo di convalida pagamento: 1,3 secondi
  - 0,77 convalide di prenotazioni al secondo
- Percentuale di utenti che prenotano soltanto il volo: $p_1 = 65\%$
- Percentuale di utenti che prenotano oltre al volo un hotel: $p_2 = 20\%$
- Percentuale di utenti che oltre all'hotel noleggiano un veicolo: $p_3 = 40\%$

4.2 Probabilità di Routing

Il flusso totale in ingresso viene distribuito verso i vari nodi tramite le **probabilità di routing**, definite sfruttando le percentuali descritte in precedenza:

- Probabilità di uscita dalla prenotazione del volo
  - Convalida Pagamento: $p_1 = 0{,}65$
  - Prenotazione Hotel: $p_2 = 0.20$
  - Prenotazione Noleggio: $1\text{-}p_1\text{-}p_2 = 0.15$
- Probabilità di uscita dalla prenotazione dell'hotel
  - Prenotazione Noleggio: $p_3 = 0{,}40$
  - Convalida Pagamento: $1\text{-}p_3 = 0.60$
- La probabilità di uscita dai nodi 3-4 è pari ad 1, questo perché la destinazione è deterministica
  - Ovviamente, c'è una percentuale p_{loss} che viene scartata dal nodo di convalida a causa della coda finita.
  - In questo caso, però, non c'è destinazione, ma la richiesta viene rigettata dal sistema.

5 Modello Computazionale

Per la simulazione della rete è stato utilizzato l'approccio di tipo **Next-Event Simulation**, in cui l'avanzamento del tempo si effettua tramite processamento dell'evento successivo.

Il modello è sviluppato in linguaggio C sfruttando le librerie di Larry Leemis descritte al link *http://www.math.wm.edu/~leemis/simtext.code.c*: in particolare si utilizzano i files *rngs.h*, *rvgs.h* e *rvms.h* rispettivamente per la generazione di numeri random, di variabili aleatorie ed intervalli di confidenza.

Il codice della simulazione è implementato nei files *microservices_base.c*, *microservices_resized.c* e *microservices_improved.c*. I files sono molto simili tra loro, ma è necessario differenziare lo scenario di base, da quello ridimensionato e da quello migliorato, che presentano configurazioni di serventi differenti.

Nella directory /lib troviamo le librerie descritte in precedenza con l'aggiunta della libreria custom *utils.h* che contiene diverse funzioni user-defined utilizzate all'interno dell'implementazione per avere un codice più ordinato.

Infine, nel file *config.h* troviamo la definizione delle strutture utilizzate all'interno della simulazione e i parametri di configurazione globale del sistema.

5.1 Strutture Dati

Descriviamo le principali strutture dati utilizzate all'interno della simulazione:

- Struttura *node_stats* che mantiene le informazioni sul singolo nodo.
    - Mantiene il riferimento ai server associati e informazioni sulle statistiche individuali (jobs in coda, jobs in servizio, jobs processati, …).
    - La coda del nodo è implementata come una *linked list* di job in cui ogni job punta a quello successivo.
- Struttura *server_stats* che mantiene le informazioni sul singolo servente.
    - Mantiene il riferimento al job in servizio (se presente) e informazioni sulle statistiche individuali (stato, jobs processati, ultima uscita, …).
- Struttura *event* che rappresenta la lista degli eventi.
    - Memorizza il tipo di evento (arrivo o fine servizio), il nodo su cui l'evento avviene, il server che deve servire il job, l'istante di tempo in cui inizia l'evento e il riferimento al prossimo evento più imminente.
- Struttura *analysis* che mantiene le statistiche medie temporanee prodotte da una singola run della simulazione (replica/batch).
- Struttura *statistic_analysis* che mantiene le statistiche medie finali prodotte dall'esecuzione dell'intera simulazione e gli intervalli di confidenza.
- Per mantenere il tempo di simulazione si usa la variabile globale *current_time.*

5.2 Gestione degli eventi

Gli eventi di arrivo e di completamento vengono gestiti in maniera differente, per questo motivo si verifica sempre se il prossimo evento è un arrivo o un completamento. Per l'implementazione della event_list, all'estrazione di un evento questo viene direttamente eliminato dalla lista:

1. Si estrae il primo evento disponibile dalla event_list tramite ExtractEvent().
2. Si controlla il tipo di evento e si processa un arrivo (*job_arrival*) o una partenza (*job_departure*) sul nodo corrispondente.

5.2.1 Arrivo

La event_list viene inizializzata generando degli eventi iniziali tramite GenerateEvent() questi sono arrivi dall'esterno con tempo di arrivo ottenuto da GetInterArrival() che utilizza la funzione Exponential() di rvgs.h.

Ogni arrivo dall'esterno viene gestito dal nodo per la prenotazione del volo o per la prenotazione dell'hotel.

Se c'è un servente libero, quindi il numero di job nel nodo è minore dei servers totali, si assegna l'evento al servente libero da più tempo tramite SelectServer():

1. Si genera un job per il processamento dell'evento di arrivo tramite GenerateJob() con tempo di servizio esponenziale tramite GetService().
2. Si registra il job nel nodo assegnandolo a un servente libero tramite SelectServer().
3. Si setta lo stato del server come BUSY e si assegna il job attualmente in servizio.
4. Si genera un evento di completamento per il servente in questione tramite GenerateEvent() e si inserisce nella event_list tramite InsertEvent().

Se non c'è alcun servente libero nel nodo, il job viene inserito nella coda del nodo, a patto che la coda sia infinita o che ci sia spazio in una coda finita.

5.2.2 Completamento

Quando viene processato un completamento, sì aggiornano le statistiche del servente in questione (jobs processati, jobs nel nodo, …) e si rimuove il job che era in servizio.

A questo punto, se ci sono jobs in coda, si estrae il primo (secondo la politica FIFO) e si mette subito in servizio, altrimenti si setta lo stato del server come IDLE.

Il completamento di un servente equivale ad un arrivo sul nodo successivo:

1. Tramite il metodo SwitchNode() si seleziona il nodo successivo che deve processare l'evento in esame secondo le probabilità di routing prima descritte.
    - Si genera un valore casuale tra 0 e 1 tramite Random() e si confronta con le diverse probabilità, individuando il nodo di destinazione.
2. Si genera un evento di arrivo sul nodo selezionato tramite GenerateEvent() e si inserisce nella event_list tramite InsertEvent().

5.3 Simulazione ad Orizzonte Infinito

Nella simulazione ad orizzonte infinito il sistema viene eseguito per un tempo di simulazione "infinito", ciò vuol dire per un tempo di gran superiore al tempo del caso di studio reale: in questo modo si producono le cosiddette statistiche a stato stazionario del sistema.

Per ricavare la media campionaria del tempo di risposta si utilizza il metodo delle **Batch Means**, che opera suddividendo l'esecuzione della simulazione in **K** batches di dimensione **B**:

- Si ricavano le statistiche tramite extract_analysis() e si resettano questi valori tramite reset_stats() per ogni singolo batch.
    - Si resettano soltanto i valori degli integrali così da reinizializzare i valori per l'esecuzione del batch successivo.
- In questo modo si genera un campione di K batches indipendenti, sul quale è possibile valutare la media campionaria.
    - Lo stato del sistema non viene alterato ma si mantengono tutti i job rimanenti dal precedente batch.
- Per gli intervalli di confidenza, si usa un livello di confidenza pari a 99%.

Le dimensioni di B e K impattano sulla qualità del campione, un valore di B grande permette di avere un campione con bassa autocorrelazione mentre un numero K più grande di batch fornisce un valore migliore in termini di intervallo di confidenza:

- Per tale motivo sono stati utilizzati K = 200 batch, valore che risulta essere un buon compromesso come riportato in letteratura (a partire da K=128).
- Il numero di jobs scelto è pari a B = 500.000, questo per avere un tempo di simulazione di all'incirca 500 giorni.
- In totale, per una run di simulazione sono processati B*K=110.000.000 jobs.

Gli obiettivi della simulazione ad orizzonte infinito per il caso di studio in esame sono principalmente tre:

- Analisi dei tempi di risposta a steady state.
- Analisi della probabilità $\mathbf{P_{loss}}$.
    - Valutata come il rapporto tra il numero di clienti che non riescono a procedere con la convalida del pagamento (scartati) ed il numero totale di arrivi nel nodo in questione
- Ricerca della configurazione ottima che permette di rispettare i due QoS e allo stesso tempo minimizzare i costi.

5.4 Simulazione ad Orizzonte Finito

Nella simulazione ad orizzonte finito il sistema viene simulato per un tempo pari a quello della singola giornata (24 ore) facendo fede al tempo del caso di studio reale: in questo modo si producono le cosiddette statistiche transienti del sistema.

Per ricavare la media campionaria del tempo di risposta si utilizzano 200 repliche della singola run di simulazione, viene quindi eseguito un ensamble di dimensione pari a 200 repliche di simulazione.:

- Ogni replica viene utilizzata per misurare le stesse statistiche da cui verranno poi ricavati i valori finali come media campionaria.
- Per gli intervalli di confidenza, si usa un livello di confidenza pari a 99%
- Il seed viene impostato tramite PlantSeeds() fuori dal ciclo di replicazione, nelle repliche successive alla prima invece viene usato come stato iniziale del generatore random lo stato finale degli streams della replica precedente [1].
    - Evitiamo sovrapposizioni degli eventi generati dalle singole repliche.
- Il termine della simulazione è legato alla fine della giornata, ovvero quando il clock di simulazione raggiunge 86400 secondi (24 ore).

L'obiettivo della simulazione ad orizzonte finito per il caso di studio in esame è principalmente uno:

- Analizzare i tempi di convergenza del sistema oppure una sua eventuale divergenza.

6 Verifica

In questo capitolo si vuole accertare che il modello computazionale è effettivamente conforme al modello delle specifiche e che i valori di output siano coerenti tra loro, quindi si vuole provare che l'implementazione del modello computazionale sia effettivamente corretta.

Per valutare queste condizioni sono state eseguite diverse run ad orizzonte sia finito che infinito, nel caso specifico riportiamo l'output dell'analisi ad orizzonte infinito basata su una simulazione suddivisa in 200 batches con il 99.00% di confidenza:

Risultati nodo 1

Tempo di interarrivo medio	0.526375 +/- 0.00016
Tempo di risposta medio	10.785229 +/- 0.1231
Tempo di attesa medio	8.785667 +/- 0.122785
Tempo di servizio medio	1.999562 +/- 0.000587
Numero di richieste medio nel nodo	20.49176 +/- 0.234577
Numero di richieste medio nella coda	16.692665 +/- 0.233852
Utilizzazione media	0.949774 +/- 0.000332
Probabilità di perdita	0.00% +/- 0.00%

Risultati nodo 2

Tempo di interarrivo medio	0.847526 +/- 0.000369
Tempo di risposta medio	15.709376 +/- 0.211155
Tempo di attesa medio	12.509269 +/- 0.210495
Tempo di servizio medio	3.200106 +/- 0.001267
Numero di richieste medio nel nodo	18.538527 +/- 0.252323
Numero di richieste medio nella coda	14.762341 +/- 0.250928
Utilizzazione media	0.944046 +/- 0.000498
Probabilità di perdita	0.00% +/- 0.00%

Risultati nodo 3

Tempo di interarrivo medio	1.321021 +/- 0.000656
Tempo di risposta medio	23.710621 +/- 0.407126
Tempo di attesa medio	21.210868 +/- 0.405354
Tempo di servizio medio	2.499753 +/- 0.001283
Numero di richieste medio nel nodo	17.951687 +/- 0.310972
Numero di richieste medio nella coda	16.059211 +/- 0.310132
Utilizzazione media	0.946238 +/- 0.000515
Probabilità di perdita	0.00% +/- 0.00%

Risultati nodo 4

Tempo di interarrivo medio	0.651407 +/- 0.000223
Tempo di risposta medio	5.669898 +/- 0.002669
Tempo di attesa medio	4.369899 +/- 0.002239
Tempo di servizio medio	1.299999 +/- 0.000451
Numero di richieste medio nel nodo	8.704879 +/- 0.002158
Numero di richieste medio nella coda	6.709013 +/- 0.001755
Utilizzazione media	0.997933 +/- 0.000272
Probabilità di perdita	43.14% +/- 0.02%

6.1 Modello delle specifiche

Per il caso di studio in esame, la verifica consiste nell'accertarsi che i flussi in arrivo ai nodi ed i tempi di servizio degli stessi, siano coerenti con quelle del modello delle specifiche, quindi che:

1. Il tempo medio di inter-arrivo su un nodo deve essere uguale all'inverso del suo flusso entrante, relativamente alle probabilità di routing dichiarate nel modello delle specifiche
2. Il tempo medio di servizio deve essere pari a quello riportato nel modello delle specifiche

Di seguito è riportata la verifica:

1. $\frac{1}{flussoentrante} \approx i$
    a. $f_1 = a_1 = 1.9 \rightarrow \frac{1}{1,9} \approx 0.526315 \approx i_1$
    b. $f_2 = a_2 + p_2 f_1 = 1.18 \rightarrow \frac{1}{1,18} \approx 0.847457 \approx i_2$
    c. $f_3 = (1 - p_1 - p_2) f_1 + p_3 f_2 = 0.757 \rightarrow \frac{1}{1,9} \approx 0.526315 \approx i_3$
    d. $f_4 = \big((1 - p|3) f_2 + p_1 f_1 + f_3\big)(1 - p_{loss}) = 1.5352 \rightarrow \frac{1}{1.5352} \approx 0.650298 \approx i_4$

2. $E(S_i) \approx specifica$
    a. $1.999562 \approx 2.0 \approx E\big(S_{1,i}\big)$
    b. $3.200106 \approx 3.2 \approx E\big(S_{2,i}\big)$
    c. $2.499753 \approx 2.5 \approx E\big(S_{3,i}\big)$
    d. $1.299999 \approx 1.3 \approx E\big(S_{4,i}\big)$

6.2 Controlli di consistenza

Gli ulteriori controlli di consistenza realizzati sono i seguenti:

1. La popolazione media in un nodo deve essere uguale alla somma tra quella in coda e quella mediamente in servizio (pari a m volte il tasso di utilizzazione)
2. Il tempo di risposta di un nodo deve essere uguale alla somma tra il tempo medio in coda ed il tempo medio di servizio
3. Deve valere la legge di Little
    a. La popolazione media in un nodo deve essere uguale al prodotto tra il tempo medio di risposta del nodo e il suo flusso entrante
    b. La popolazione media in coda di un nodo deve essere uguale al prodotto tra il tempo medio in coda del nodo e il suo flusso entrante

Di seguito è riportata la verifica:

1. $E(N_Q) + E(c) \approx E(N_S)$
    - $16.692665 + 4 * 0949774 = 20.491761 \approx E(N_{S,1})$
    - $14.762341 + 4 * 0.944046 = 18.538526 \approx E(N_{S,2})$
    - $16.059211 + 2 * 0.946238 = 17.951687 \approx E(N_{S,3})$
    - $6.7090130 + 2 * 0.997933 = 8.7048790 \approx E(N_{S,4})$

2. $E(T_Q) + E(S_i) \approx E(T_S)$
    - $8.785667 + 1.999562 = 10.785229 \approx E(T_{S,1})$
    - $12.509269 + 3.200106 = 15.709375 \approx E(T_{S,2})$
    - $21.210868 + 2.499753 = 23.71062 \approx E(T_{S,3})$
    - $4.369899 + 1.299999 = 5.669898 \approx E(T_{S,4})$

3. $N = \lambda T$
    a. $E(N_S) = f * E(T_S) = \frac{E(T_S)}{i}$
        - $10.785229/0.526375 = 20.489630 \approx E(N_{S,1})$
        - $15.709376/0.847526 = 18.535568 \approx E(N_{S,2})$
        - $23.710621/1.321021 = 17.948709 \approx E(N_{S,3})$
        - $5.669898/0.651407 = 8.704079 \approx E(N_{S,4})$

b. $E(N_Q) = f * E(T_Q) = \frac{E(T_Q)}{i}$

- $8.785667/0.526375 = 8.704079 \approx E(N_{Q,1})$
- $12.509269/0.847526 = 8.704079 \approx E(N_{Q,2})$
- $21.210868/1.321021 = 8.704079 \approx E(N_{Q,3})$
- $4.369899/0.651407 = 8.704079 \approx E(N_{Q,4})$

7 Validazione

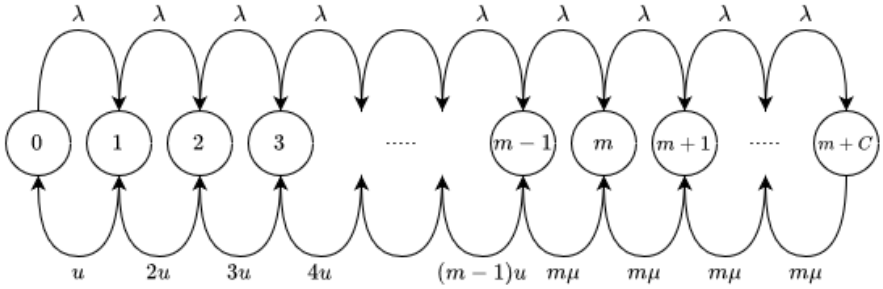
In questa fase si vuole accertare che il modello computazionale è coerente con il sistema reale in esame, tuttavia, non avendo dati reali a disposizione, si opera una validazione rispetto al modello analitico, quindi si verifica che i risultati ottenuti dalla simulazione rispettino effettivamente le leggi teoriche.

Come per la verifica, anche la validazione del sistema, e quindi delle leggi teoriche, è stata eseguita sull'output della simulazione ad orizzonte infinito, per questo motivo sono riportati i risultati di questa run rispetto al modello analitico.

Le leggi del modello analitico che caratterizzano i nodi **M/M/m** del sistema sono:

- $E(T_{Q,k}) = P_{Q,k}\frac{E(S,k)}{1-\rho_k}$ $\quad per\ i\ nodi\ con\ k \in [1,3]$
  - $P_{Q,k} = \frac{(m|k\rho_k)^{m_k}}{m_k!(1-\rho_k)}P_k(0)$
  - $P_k(0) = \left[\sum_{i=0}^{m_k-1}\frac{(m_k\rho_k)^i}{i!} + \frac{(m_k\rho_k)^{m_k}}{m_k!(1-\rho_k)}\right]^{-1}$
- $E(T_{S,k}) = E(T_{Q,k}) + E(S_{k,i})$ $\quad per\ i\ nodi\ con\ k \in [1,3]$
- $E(N_{Q,k}) = \lambda_k E(T_{Q,k})$ $\quad per\ i\ nodi\ con\ k \in [1,3]$
- $E(N_{S,k}) = \lambda_k E(T_{S,k})$ $\quad per\ i\ nodi\ con\ k \in [1,3]$

Per quanto riguarda il nodo **M/M/m/C**, invece, lo studio risulta essere più complesso perché è necessario derivare le formule che modellano un multi-server con capacità finita della coda analizzando la rispettiva catena di Markov:



- Per prima cosa scriviamo le equazioni di *bilanciamento globale*, il flusso che entra nello stato S_i è uguale al flusso che esce dallo stato S_i
  - $(\lambda + (m-1)\mu)\pi_{m-1} = \pi_{m-2}\lambda + m\mu\pi_m \rightarrow \pi_m = \frac{1}{m!}\frac{\lambda^m}{\mu^m}\pi_0$
  - $(\lambda + m\mu)\pi_m = \pi_{m-1}\lambda + m\mu\pi_{m+1} \rightarrow \pi_{m+1} = \frac{1}{mm!}\frac{\lambda^{m+1}}{\mu^{m+1}}\pi_0$
  - $(\lambda + m\mu)\pi_{m+1} = \pi_m\lambda + m\mu\pi_{m+2} \rightarrow \pi_{m+2} = \frac{1}{m^2m!}\frac{\lambda^{m+2}}{\mu^{m+2}}\pi_0$
  - Si ottiene che $\pi_k = \begin{cases}\frac{1}{k!}(m\rho)^k\pi_0, \wedge\ k \le m \\ \frac{m^m}{m!}\rho^k\pi_0, \wedge\ k > m\end{cases}$ $dove \rho = \frac{\lambda}{m\mu}$

- Passiamo ora alla *condizione di normalizzazione* per ricavare π_0
  - $\sum_{i=0}^{m+C} \pi_i = 1 \rightarrow \left(\sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} + \frac{m^m}{m!}\sum_{m}^{m+C} \rho^k\right)\pi_0 = 1$
  - Si ottiene che $\pi_0 = \frac{1}{\sum_{k=0}^{m-1}\frac{(m\rho)^k}{k!} + \frac{m^m}{m!}\sum_{m}^{m+C}\rho^k} = \frac{1}{\sum_{k=0}^{m-1}\frac{(m\rho)^k}{k!} + \frac{(m\rho)^m}{m!}\sum_{k=0}^{C}\rho^k}$
- Deriviamo p_{loss}, $E(N_{S,4})$ ed $E(N_{Q,4})$
  - $p_{loss} = \pi_{m+C} = \frac{\frac{(m\rho)^m}{m!}\rho^C}{\sum_{k=0}^{m-1}\frac{(m\rho)^k}{k!} + \frac{(m\rho)^m}{m!}\frac{1-\rho^{C+1}}{1-\rho}}$
  - $E(N_{S,4}) = \sum_{k=0}^{m+C} k\pi_k = \frac{\sum_{k=0}^{m-1} k\frac{(m\rho)^k}{k!}\rho^C + \sum_{k=m}^{m+C} k\frac{m^m}{m!}\rho^k}{\sum_{k=0}^{m-1}\frac{(m\rho)^k}{k!} + \frac{(m\rho)^m}{m!}\frac{1-\rho^{C+1}}{1-\rho}}$
  - $E(N_{Q,4}) = E(N_{S,4}) - m\rho' = E(N_{S,4}) - m\rho(1 - p_{loss})$
- Deriviamo $E(T_{Q,4})$ ed $E(T_{S,4})$
  - Si applica la Legge di Little $N = \lambda T$, nel caso specifico in analisi si ha che $\lambda_4 = f_4(1 - p_{loss}) = (p_1 f_1 + (1 - p_3)f_2 + f_3)(1 - p_{loss})$
  - $E(T_{Q,4}) = \frac{E(N_{Q,4})}{\lambda_4} = \frac{E(N_{S,4}) - m\rho(1-p_{loss})}{((1-p_3)f_2 + p_1 f_1 + f_3)(1-p_{loss})}$
  - $E(T_{S,4}) = \frac{E(N_{S,4})}{\lambda_4} = \frac{E(N_{S,4})}{((1-p_3)f_2 + p_1 f_1 + f_3)(1-p_{loss})}$

A questo punto, abbiamo a disposizione tutte le leggi teoriche necessarie per verificare che effettivamente il modello computazionale è coerente con il modello analitico:

	Modello Analitico	**Modello Sperimentale** ($\alpha = 0.01$)
$E(N_{S,1})$	20.736960908382	20.491760 +/- 0.234577
$E(N_{S,2})$	18.588599371818	18.538527 +/- 0.252323
$E(N_{S,3})$	18.0908425565298	17.951687 +/- 0.310972
$E(N_{S,4})$	8.70668118927952	8.704879 +/- 0.002158
$E(T_{Q,1})$	8.91418995178	8.785667 +/- 0.122785
$E(T_{Q,2})$	12.5530503151	12.509269 +/- 0.210495
$E(T_{Q,3})$	21.3980747114	21.210868 +/- 0.406364
$E(T_{Q,4})$	4.37035814768	4.369899 +/- 0.002239
p_{loss}	43.142107411 %	43.1375 % +/- 0.0248 %
$E(T_{S,tot})$	56.236731259	55.8751 +/- 0.5335

Con $E(T_{S,tot})$ si fa riferimento al massimo tempo medio di risposta del sistema, tempo che impiega una richiesta generica per completare il percorso che attraversa tutti quanti i nodi del sistema:

- Prenotazione e convalida di un pacchetto vacanza con hotel, aereo e noleggio
- $E(T_{S,tot}) = E(T_{S,1}) + E(T_{S,2}) + E(T_{S,3}) + E(T_{S,4}) = \sum_k \left(E(T_{Q,k}) + E(S_{k,i})\right)$
- Diverso dal tempo medio di risposta globale del sistema in una rete di Jackson, valore che non considera un percorso specifico per i jobs: $E(t_r) = \sum_k v_k E(t_k)$

8 Analisi dei Risultati

Per osservare graficamente le funzioni che descrivono gli indici di prestazione del modello analitico al variare dell'utilizzazione e del numero di serventi, si è fatto uso del generatore online di grafici di funzione Desmos, accessibile ai seguenti links:

- Microservices base
- Microservices_resized
- Microservices_improved

In particolare, di seguito si propongono i grafici del modello analitico relativi agli indici prestazionali del nodo con coda limitata:

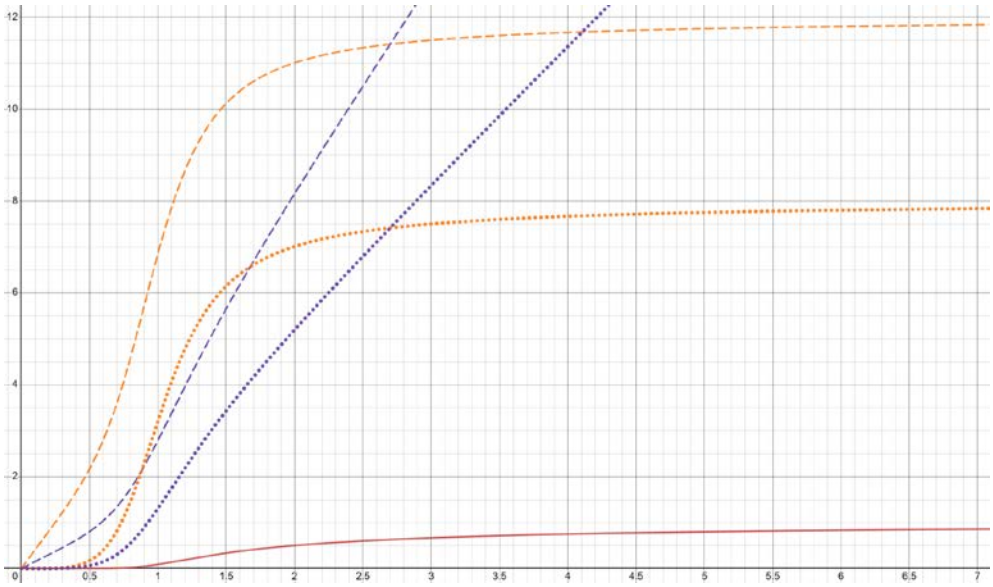
$$p_{loss4}(x) = \frac{\frac{(m_4 x)^{m_4} x^{C_4}}{m_4!}}{\left(\sum_{k=0}^{m_4-1} \frac{(m_4 x)^k}{k!}\right) + \frac{(m_4 x)^{m_4}}{m_4!} \cdot \frac{1-x^{C_4+1}}{1-x}} \{x \geq 0\}$$

$$N_{S4}(x) = \frac{\sum_{k=0}^{m_4-1} k\frac{(m_4 x)^k}{k!} + \frac{m_4^{m_4}}{m_4!}\sum_{k=m_4}^{m_4+C_4} kx^k}{\sum_{k=0}^{m_4-1} \frac{(m_4 x)^k}{k!} + \frac{(m_4 x)^{m_4}}{m_4!} \cdot \frac{1-x^{C_4+1}}{1-x}} \{x \geq 0\}$$

$$T_{S4}(x) = \frac{N_{S4}(x)}{f_4(1-p_{loss4}(x))} \{x \geq 0\}$$

$$N_{Q4}(x) = N_{S4}(x) - m_4 x(1-p_{loss4}(x)) \{x \geq 0\}$$

$$T_{Q4}(x) = \frac{N_{Q4}(x)}{f_4(1-p_{loss4}(x))} \{x \geq 0\}$$



Dai grafici si vede come il numero di jobs in coda, all'aumentare dell'utilizzazione, converga a C = 8, quello dei jobs nel sistema a m_4+ C = 12 e la p_{loss} a 1.

8.1 Orizzonte Infinito

L'approccio utilizzato è quello di simulare ad orizzonte infinito per identificare la configurazione ottima e per verificare che i tempi di risposta del modello siano conformi a quelli analitici.

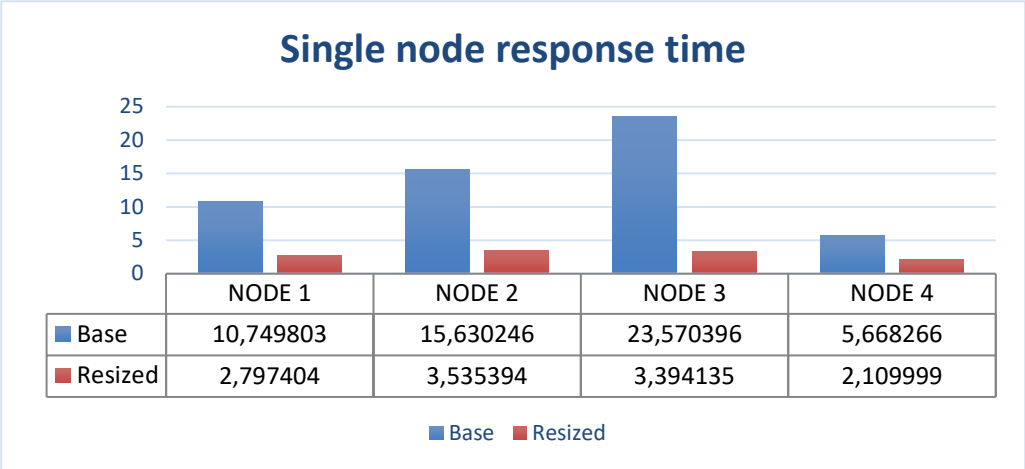
L'idea alla base della ricerca della configurazione ottima è la seguente:

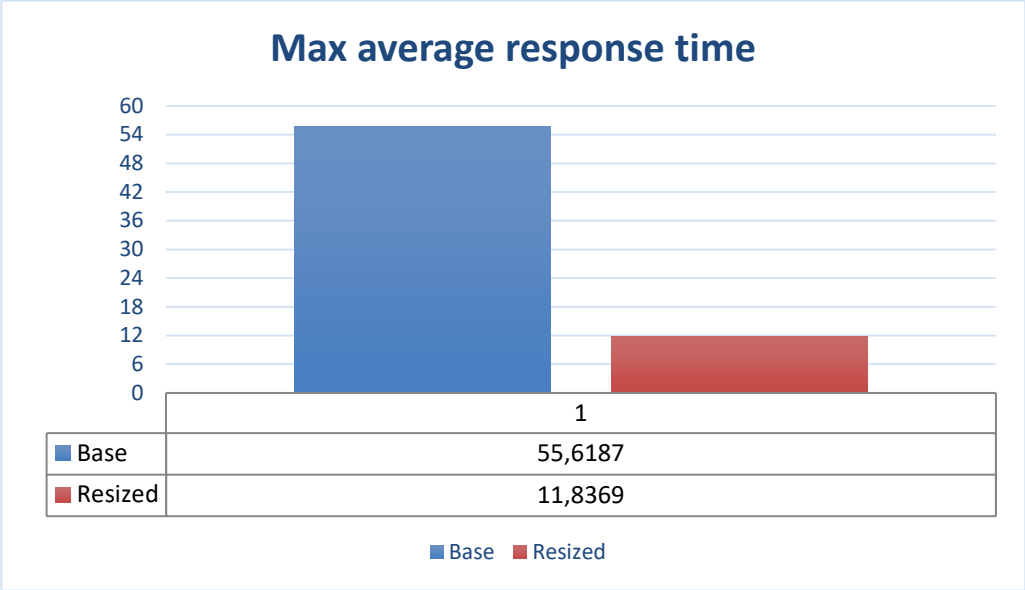
1. Aumentare il numero di serventi del nodo di convalida del pagamento fino a quando il QoS della probabilità di perdita non viene rispettato
    - Tutti i nodi sono stabili ($\rho < 1$) quindi il flusso entrante sul nodo di convalida rimane invariato anche se si migliorano gli altri nodi.
    - Ciò significa che la probabilità di perdita è funzione solamente del numero di serventi del nodo di convalida e della capacità della coda.
    - Infatti, un miglioramento sugli altri nodi porterebbe soltanto a una riduzione della loro utilizzazione.
2. Procedere con un algoritmo greedy fino a quando il QoS del massimo tempo di risposta non viene rispettato
    - Si effettuano molteplici simulazioni e ad ognuna di esse viene aggiunto un ulteriore servente al nodo che ne trarrebbe più vantaggio.
    - In particolare, si aggiunge un servente al nodo che permette di ridurre di più il massimo tempo di risposta del sistema.

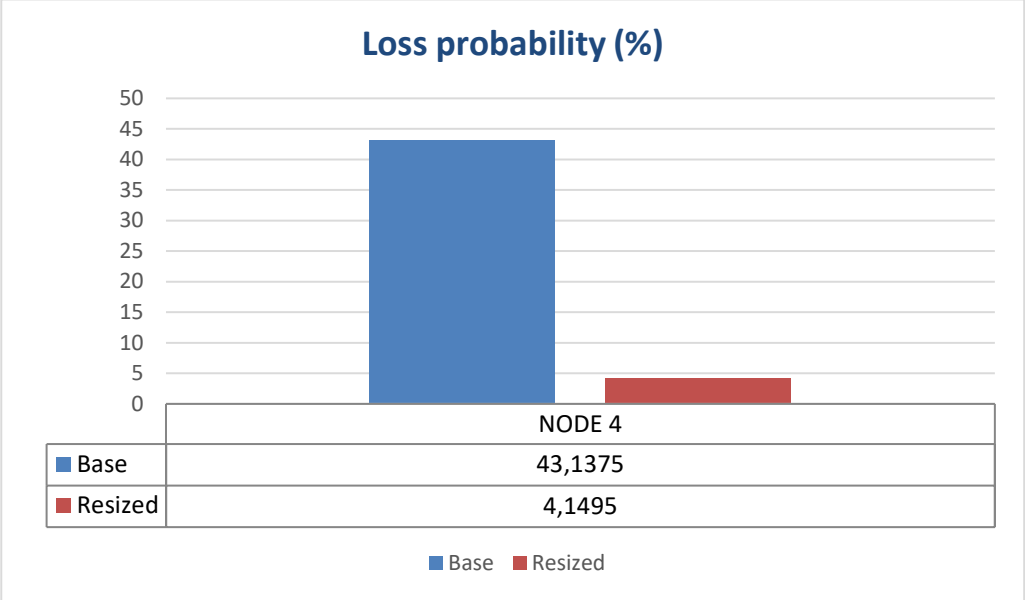
Alla fine di quest'analisi empirica, è stata individuata la configurazione ottima che permette di minimizzare il costo totale rispettando entrambi i vincoli prefissati:

- $m_1 = 5$
- $m_2 = 6$
- $m_3 = 3$
- $m_4 = 4$

Di seguito mostriamo il confronto prestazionale tra la configurazione di base del sistema e la nuova configurazione individuata, da cui possiamo riscontrare che entrambi i QoS sono rispettati:




Max average response time
60
54
48
42
36
30
24
18
12
6
0
1
Base
55,6187
Resized
11,8369
Base
Resized


Loss probability (%)
50
45
40
35
30
25
20
15
10
5
0
NODE 4
Base
43,1375
Resized
4,1495
Base
Resized

9.2 Orizzonte Finito

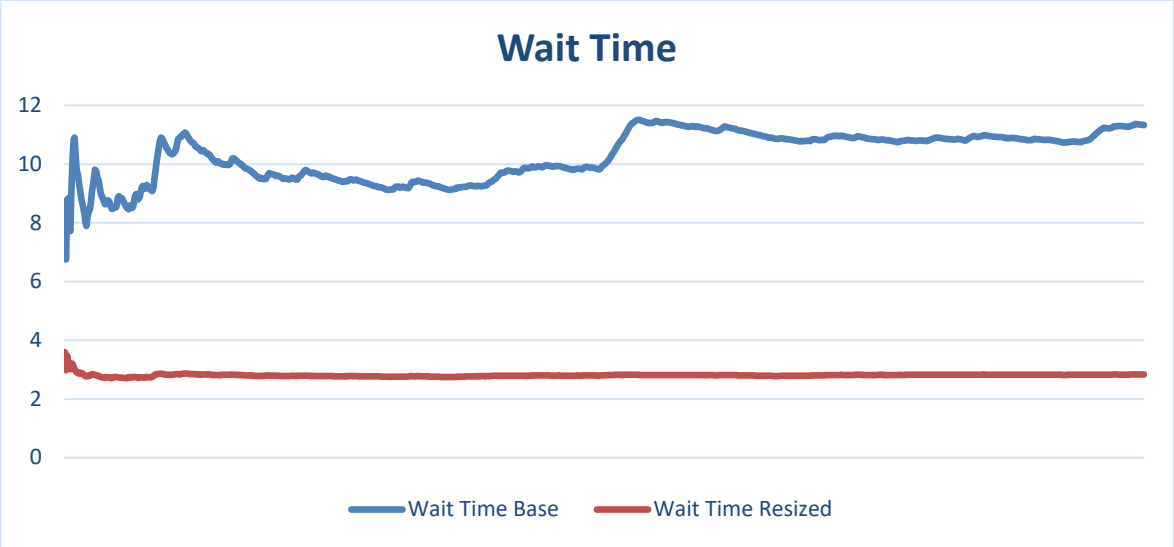
Come già detto in precedenza, l'analisi ad orizzonte finito non è necessaria ai fini del ridimensionamento, ma risulta interessante per effettuare un'analisi dettagliata del comportamento del sistema nelle prime 24 ore di utilizzo:

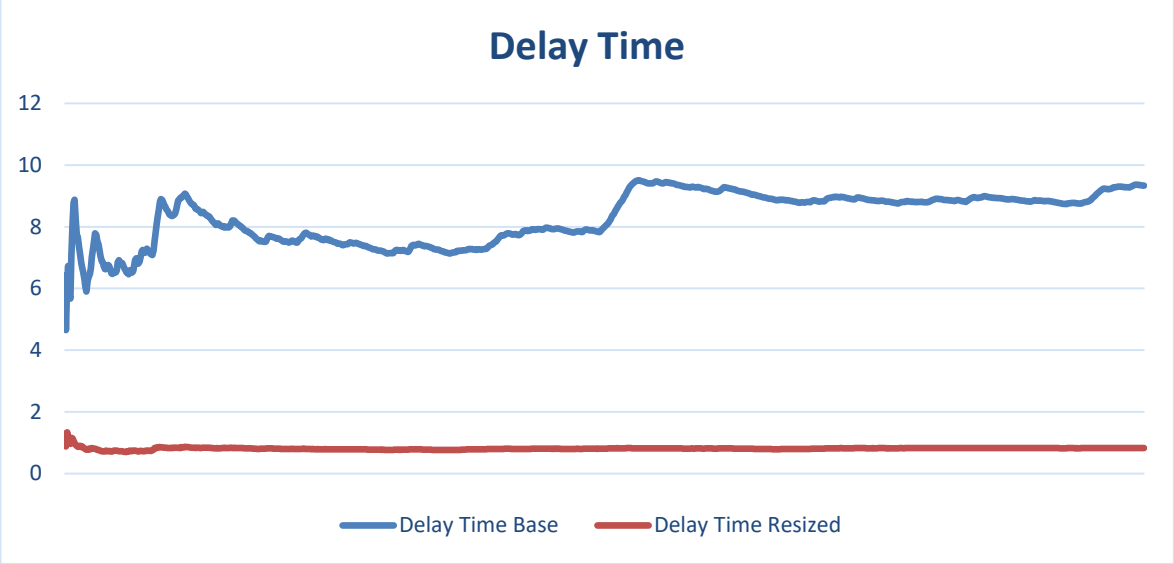
- Verificare la stabilità dei nodi
- Analizzare i tempi di convergenza

Nella seguente analisi ci si sofferma sui tempi in coda e di risposta, per gli altri indici di prestazione si prega la visione dei file excel nella cartella */other/time_trend_analysis*.

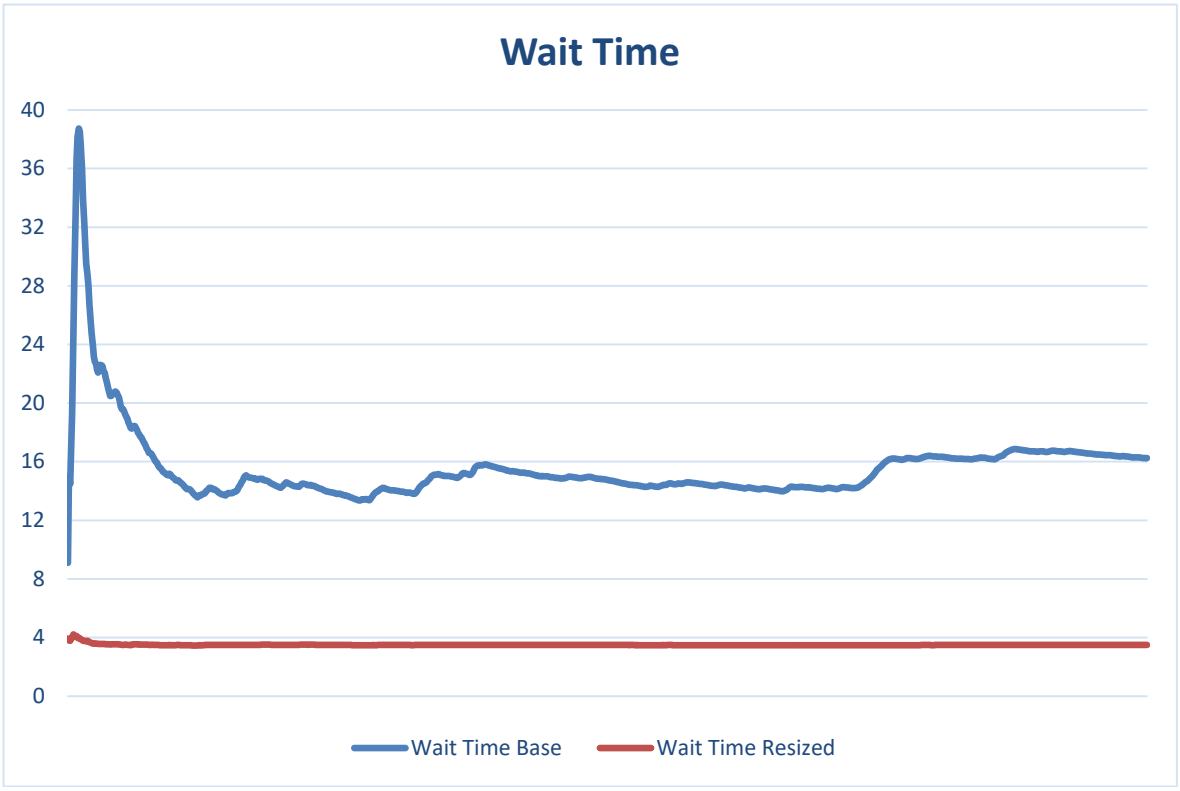
Lo studio dell'andamento temporale è stato prodotto per entrambe le configurazioni, così da poter verificare se e quando gli indici prestazionali del sistema convergono al valore atteso:

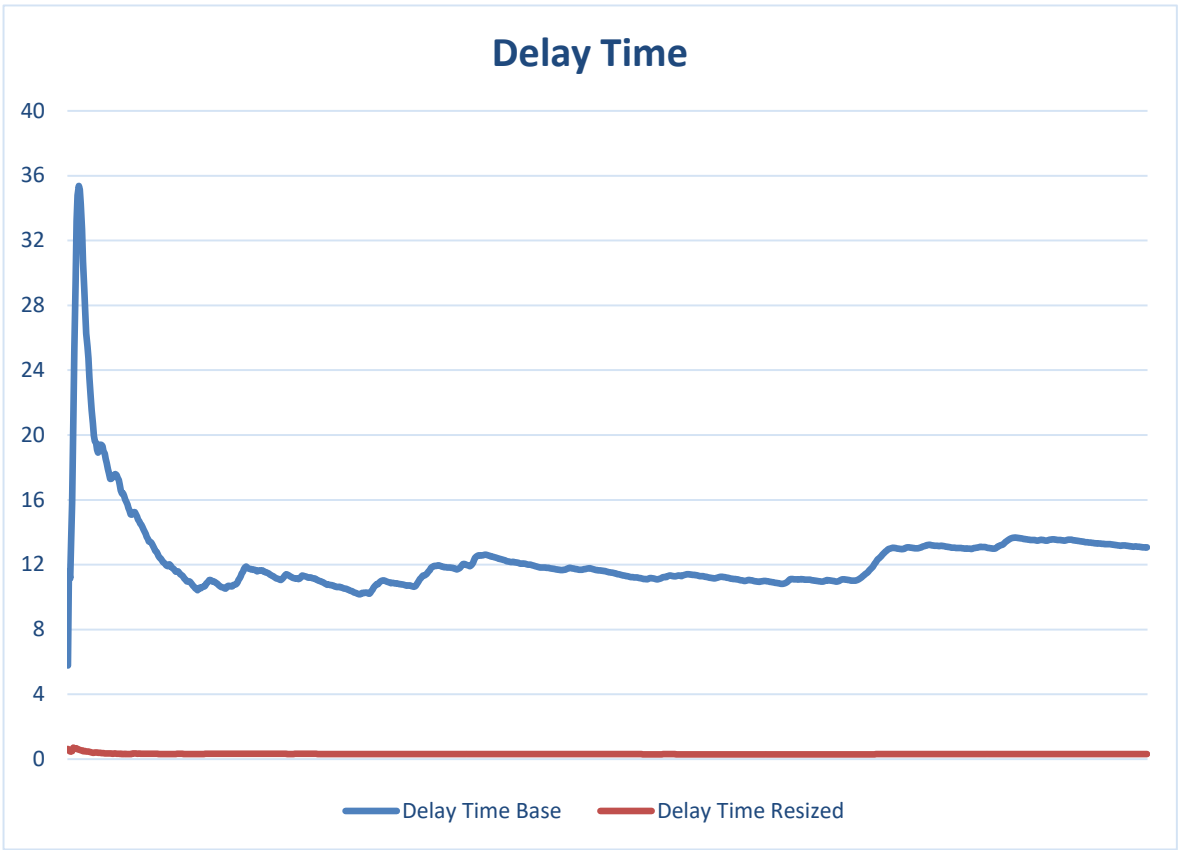
- ***Nodo di prenotazione volo***



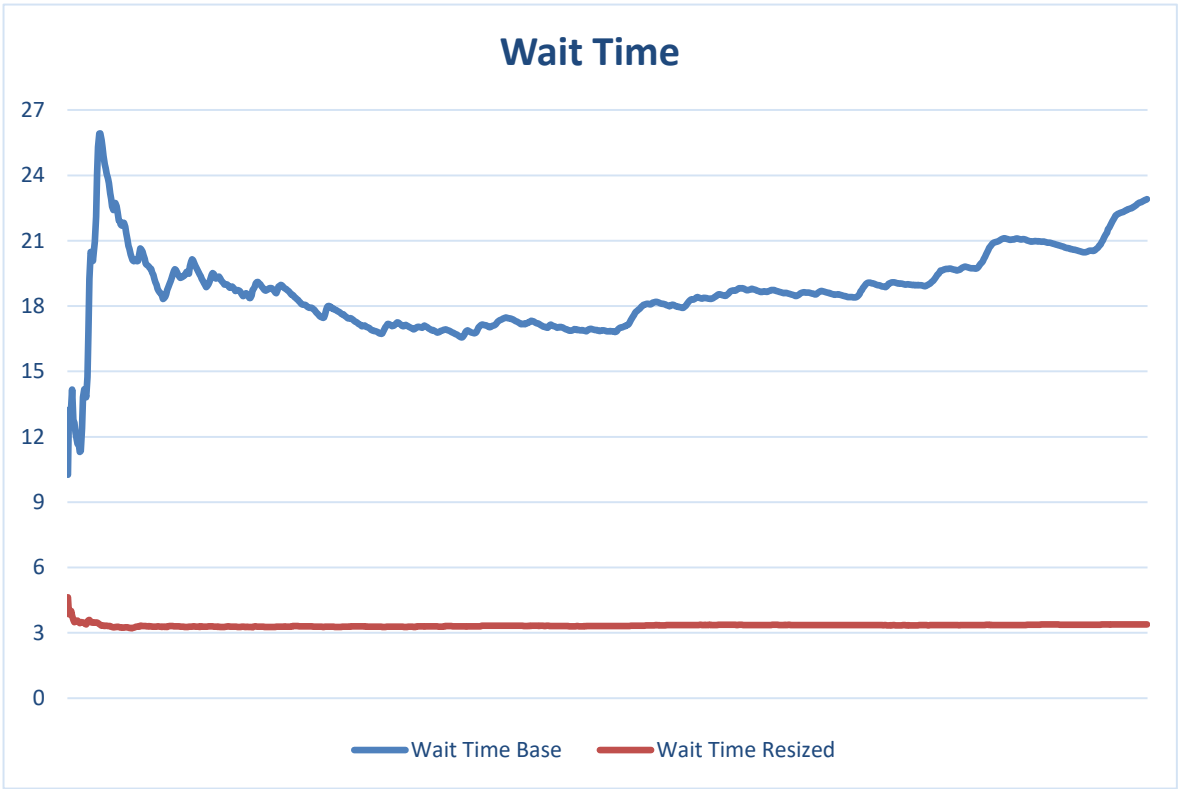


- *Nodo di prenotazione hotel*

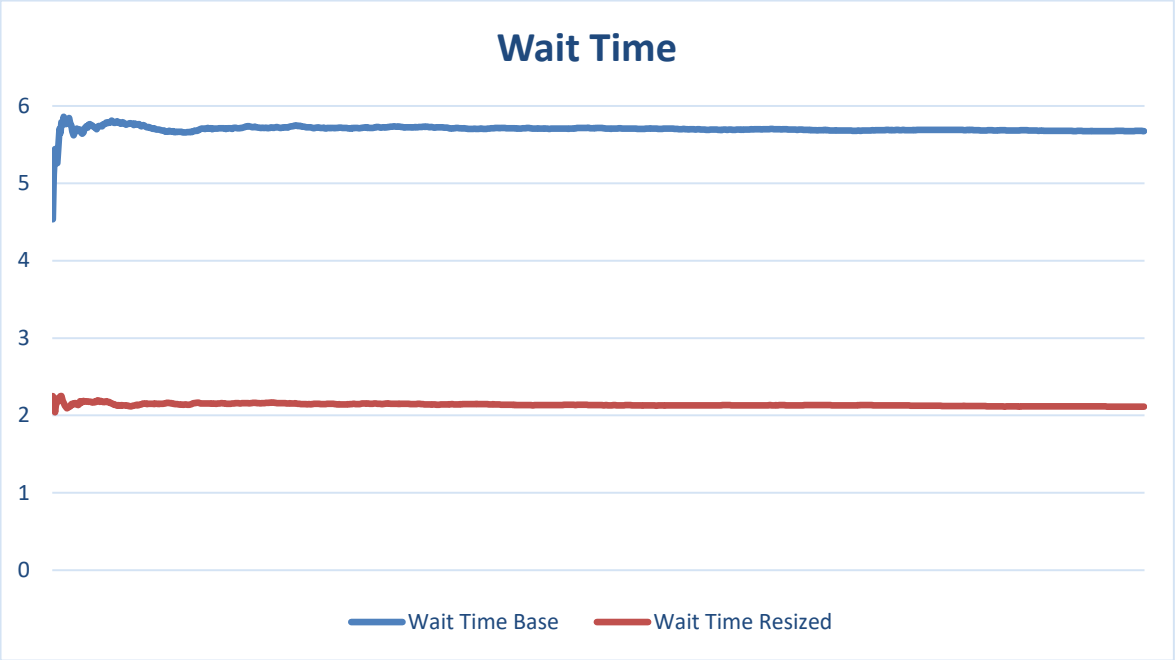


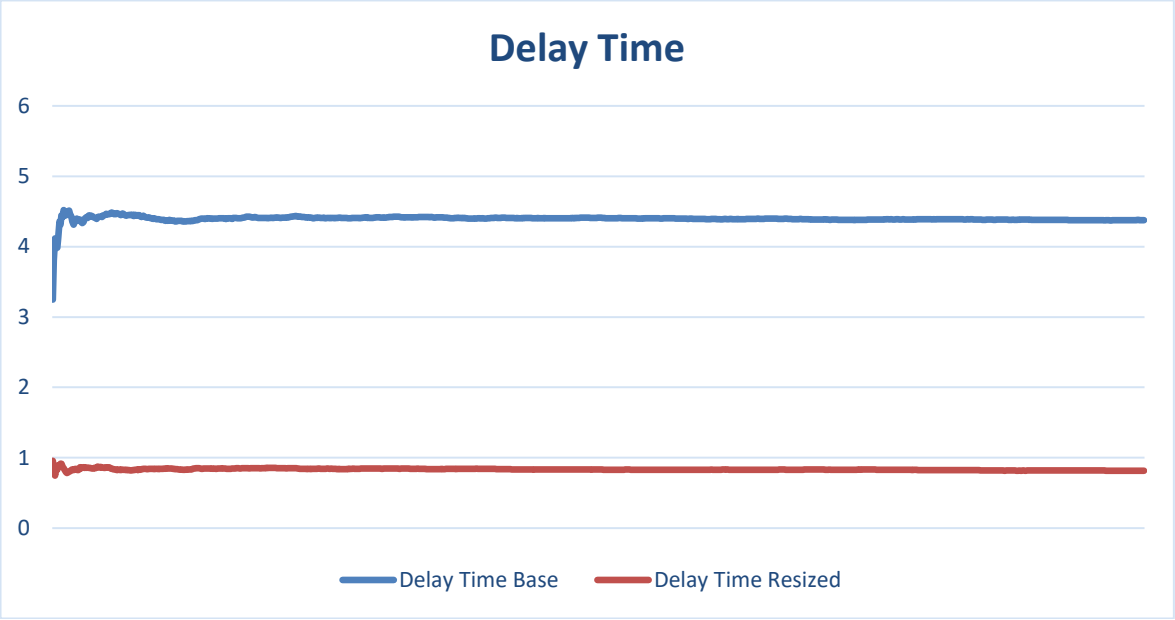


- *Nodo di prenotazione noleggio automobili*



- *Nodo di convalida pagamento*





Come possiamo osservare dai grafici riportati:

- Il sistema iniziale, a causa di tassi di utilizzazione molto alti, converge molto lentamente e presenta forte variabilità a causa dell'elevata instabilità dei nodi
    - Improvvisi aumenti e oscillazioni ampie intorno al valore atteso.
- La configurazione ottima individuata, al contrario, offre molta più stabilità e tempi di convergenza estremamente inferiori.
    - Stimata tra le due e le tre ore dall'attivazione del sistema.

9 Algoritmo Migliorativo

Come già accennato, il sistema è stato progettato con una coda limitata sul nodo finale di convalida del pagamento, questo per garantirne la stabilità ed evitare così ulteriori costi di sviluppo rappresentati dall'aggiunta di ulteriori serventi.

Tuttavia, anche se il QoS sulla probabilità di perdita è rispettato, una certa quantità di richieste scartate, seppur piccola, rappresenta comunque una perdita economica:

- La soluzione più logica per massimizzare il profitto sarebbe quella di introdurre una coda infinita sul nodo di convalida.
- Questo però potrebbe portare ad un aumento dei tempi di risposta del nodo in questione a tal punto da violare il QoS sul massimo tempo medio di risposta.

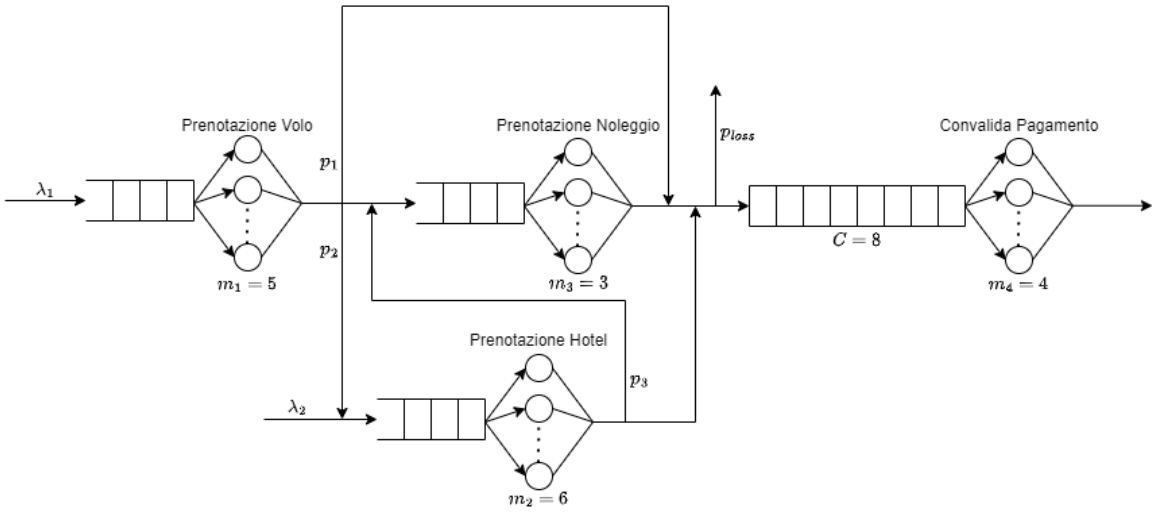
Lo scopo dell'algoritmo migliorativo, quindi, per quanto detto in precedenza, evolve nel modo seguente:

- Verificare l'effettiva violazione del QoS sul massimo tempo medio di risposta.
- Identificare un compromesso che permetta ad una porzione significativa degli utenti del sistema di rispettarlo con l'introduzione di code di priorità infinite.

Il sistema in esame differenzia tra *utenti visitatori* e *utenti iscritti*, questi ultimi rappresentano utenti che hanno fatto più volte uso del sistema di prenotazione, per questo motivo il miglioramento punta ad eliminare la perdita di richieste e allo stesso tempo soddisfare il QoS sul massimo tempo medio di risposta per gli utenti iscritti:

- Accettiamo di impiegare più tempo rispetto all'obiettivo iniziale per richieste di utenti visitatori, che sono considerate come richieste secondarie.
- Di contro, nessuna richiesta verrà persa, massimizzando il guadagno aziendale.

Il modello concettuale del sistema ridimensionato, da cui si parte per il nuovo studio su un possibile miglioramento, è il seguente:



9.1 Modello Computazionale

Il modello computazionale non presenta molte differenze con il sistema iniziale e quello ridimensionato, l'unica differenza è relativa alla gestione differente del nodo di convalida che implementa due code di priorità:

- La funzione InsertJob(), utilizzata per inserire un job nella coda di un nodo specifico, è stata modificata per gestire la priorità.
- Un job viene sempre inserito in una coda singola ma che viene gestita come due code di priorità, considerando per l'inserimento anche il livello di priorità del job oltre all'istante di tempo in cui deve verificarsi.
- La parte iniziale della coda presenta job di prima classe ordinati per tempo di arrivo, la parte finale mantiene job di seconda classe ordinati allo stesso modo.
- La coda non è divisa a metà, ma il punto di separazione dipende dalla percentuale di jobs di entrambe le classi ed evolte a runtime in base agli arrivi.

In questa maniera, saranno serviti prima tutti i jobs con priorità più alta e poi quelli con priorità più bassa, entrambi secondo una schedulazione FIFO.

9.2 Verifica

Per accertarsi che il modello computazionale è effettivamente conforme al modello delle specifiche e che i valori di output siano coerenti tra loro, sono state eseguite diverse run ad orizzonte sia finito che infinito, ma in questo caso la verifica si basa sul sistema ridimensionato in precedenza e non sulla configurazione iniziale.

Nel caso specifico è riportato l'output dell'analisi ad orizzonte infinito basata su una simulazione suddivisa in 200 batches con il 99.00% di confidenza:

Risultati nodo 1

Tempo di interarrivo medio	0.526375 +/- 0.000161
Tempo di risposta medio	2.797558 +/- 0.003752
Tempo di attesa medio	0.797996 +/- 0.003431
Tempo di servizio medio	1.999562 +/- 0.000586
Numero di richieste medio nel nodo	5.315266 +/- 0.007328
Numero di richieste medio nella coda	1.516170 +/- 0.006566
Utilizzazione media	0.759819 +/- 0.000265
Probabilità di perdita	0.00% +/- 0.00%

Risultati nodo 2

Tempo di interarrivo medio	0.847475 +/- 0.000368
Tempo di risposta medio	3.535285 +/- 0.002669
Tempo di attesa medio	0.335181+/- 0.001807
Tempo di servizio medio	3.200103 +/- 0.001276
Numero di richieste medio nel nodo	4.171956 +/- 0.003566
Numero di richieste medio nella coda	0.395549 +/- 0.002165
Utilizzazione media	0.629401 +/- 0.000309
Probabilità di perdita	0.00% +/- 0.00%

Risultati nodo 3

Tempo di interarrivo medio	1.320848 +/- 0.000709
Tempo di risposta medio	3.389750 +/- 0.005325
Tempo di attesa medio	0.890000 +/- 0.004459
Tempo di servizio medio	2.499750 +/- 0.001276
Numero di richieste medio nel nodo	2.566611 +/- 0.004430
Numero di richieste medio nella coda	0.673888 +/- 0.003466
Utilizzazione media	0.630908 +/- 0.000433
Probabilità di perdita	0.00% +/- 0.00%

Risultati nodo 4

Tempo di interarrivo medio	0.386441 +/- 0.000087
Tempo di risposta medio	2.110415 +/- 0.001475
Tempo di attesa medio	0.810357 +/- 0.001197
Tempo di servizio medio	1.300058 +/- 0.000365
Numero di richieste medio nel nodo	5.461662 +/- 0.003259
Numero di richieste medio nella coda	2.097162 +/- 0.002862
Utilizzazione media	0.841125 +/- 0.000212
Probabilità di perdita	4.15% +/- 0.01%

9.2.1 Modello delle specifiche

La verifica consiste nell'accertarsi che i flussi in arrivo ai nodi ed i tempi di servizio degli stessi, siano coerenti con quelli del modello delle specifiche prodotto dal sistema ridimensionato, che rimane del tutto invariato dal sistema iniziale:

1. Il tempo medio di inter-arrivo su un nodo deve essere uguale all'inverso del suo flusso entrante, relativamente alle probabilità di routing dichiarate nel modello delle specifiche
2. Il tempo medio di servizio deve essere pari a quello riportato nel modello delle specifiche

Di seguito è riportata la verifica:

1. $\frac{1}{flussoentrante} \approx i$
    a. $f_1 = a_1 = 1.9 \rightarrow \frac{1}{1,9} \approx 0.526315 \approx i_1$
    b. $f_2 = a_2 + p_2 f_1 = 1.18 \rightarrow \frac{1}{1,18} \approx 0.847457 \approx i_2$
    c. $f_3 = (1 - p_1 - p_2) f_1 + p_3 f_2 = 0.757 \rightarrow \frac{1}{1,9} \approx 0.526315 \approx i_3$
    d. $f_4 = \big((1 - p|3) f_2 + p_1 f_1 + f_3\big)(1 - p_{loss}) = 2.587963 \rightarrow \frac{1}{2.587963} \approx 0{,}386404 \approx i_4$
2. $E(S_i) \approx specifica$
    a. $1.999562 \approx 2.0 \approx E\big(S_{1,i}\big)$
    b. $3.200103 \approx 3.2 \approx E\big(S_{2,i}\big)$
    c. $2.499750 \approx 2.5 \approx E\big(S_{3,i}\big)$
    d. $1.300058 \approx 1.3 \approx E\big(S_{4,i}\big)$

9.2.2 Controlli di consistenza

Gli ulteriori controlli di consistenza realizzati sono i seguenti:

1. La popolazione media in un nodo deve essere uguale alla somma tra quella in coda e quella mediamente in servizio (pari a m volte il tasso di utilizzazione)
2. Il tempo di risposta di un nodo deve essere uguale alla somma tra il tempo medio in coda ed il tempo medio di servizio
3. Deve valere la legge di Little
    a. La popolazione media in un nodo deve essere uguale al prodotto tra il tempo medio di risposta del nodo e il suo flusso entrante
    b. La popolazione media in coda di un nodo deve essere uguale al prodotto tra il tempo medio in coda del nodo e il suo flusso entrante

Di seguito è riportata la verifica:

1. $E(N_Q) + E(c) \approx E(N|S)$
   - $1.516170 + 5 * 0.759819 = 5.315265 \approx E(N_{S,1})$
   - $0.395549 + 6 * 0.629401 = 4.171955 \approx E(N_{S,2})$
   - $0.673888 + 3 * 0.630908 = 2.566612 \approx E(N_{S,3})$
   - $2.097162 + 4 * 0.841125 = 5.461662 \approx E(N_{S,4})$

2. $E(T_Q) + E(S_i) \approx E(T_S)$
   a. $0.797996 + 1.999562 = 2.797558 \approx E(T_{S,1})$
   b. $0.335181 + 3.200103 = 3.535284 \approx E(T_{S,2})$
   c. $0.890000 + 2.499750 = 3.389750 \approx E(T_{S,3})$
   d. $0.810357 + 1.300058 = 2.110415 \approx E(T_{S,4})$
3. $N = \lambda T$
   c. $E(N_S) = f * E(T|s) = \frac{E(T_S)}{i}$
      - $2.797558/0.526375 = 5.314762 \approx E(N_{S,1})$
      - $3.535285/0.847475 = 4.171550 \approx E(N_{S,2})$
      - $3.389750/1.320848 = 2.566344 \approx E(N_{S,3})$
      - $2.110415/0.386441 = 5.461157 \approx E(N_{S,4})$
   d. $E(N_Q) = f * E(T|Q) = \frac{E(T|Q)}{i}$
      - $0.797996/0.526375 = 1.516022 \approx E(N_{Q,1})$
      - $0.335181/0.847475 = 0.395505 \approx E(N_{Q,2})$
      - $0.890000/1.320848 = 0{,}673810 \approx E(N_{Q,3})$
      - $0.810357/0.386441 = 2{,}096975 \approx E(N_{Q,4})$

9.3 Validazione

In questa fase si vuole accertare che il modello computazionale è coerente con il sistema reale in esame, ma come già detto, non avendo dati reali a disposizione, si opera una validazione rispetto al modello analitico sul sistema ridimensionato.

Le leggi del modello analitico che caratterizzano i nodi del sistema *sono le stesse descritte nel capitolo 7*, quindi verifichiamo che effettivamente il modello computazionale è coerente con il modello analitico:

	Modello Analitico	**Modello Sperimentale** ($\alpha = 0.01$)
$E(N_{S,1})$	5.3187147868627	5.315266 +/- 0.007328
$E(N_{S,2})$	4.17165922544946	4.171956 +/- 0.003566
$E(N_{S,3})$	2.567799278676707	2.566611 +/- 0.004430
$E(N_{S,4})$	5.46083122987232	5.461662 +/- 0.003259
$E(T_{Q,1})$	0.799323572033	0.797996 +/- 0.003431
$E(T_{Q,2})$	0.335304428347	0.335181 +/- 0.001807
$E(T_{Q,3})$	0.892073023351	0.890000 +/- 0.004459
$E(T_{Q,4})$	0,81066019671	0.810357 +/- 0.001197
p_{loss}	4.14954715564 %	4.1495 % +/- 0.0142 %
$E(T_{S,tot})$	11.8373612204	11.833000 +/- 0.007700

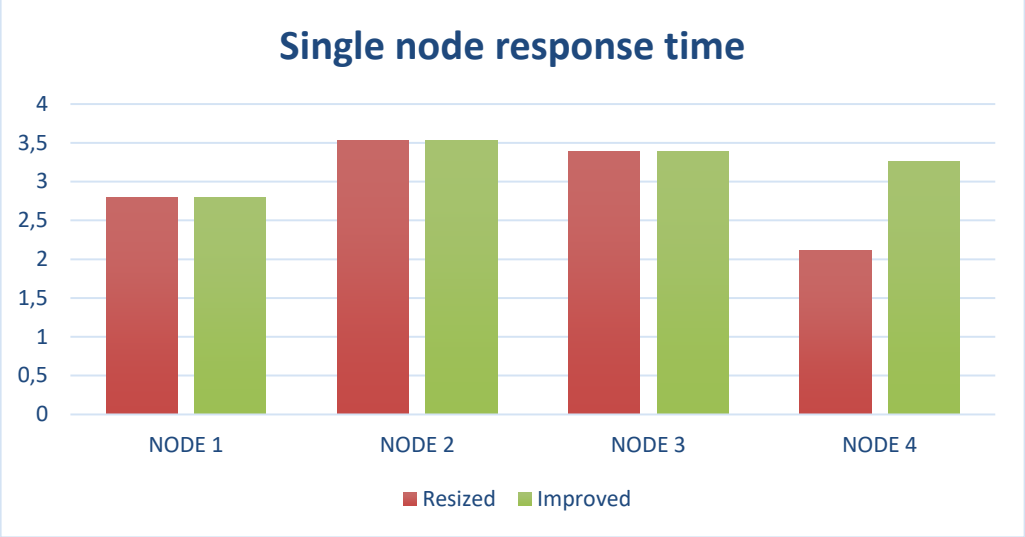
9.4 Analisi dei Risultati

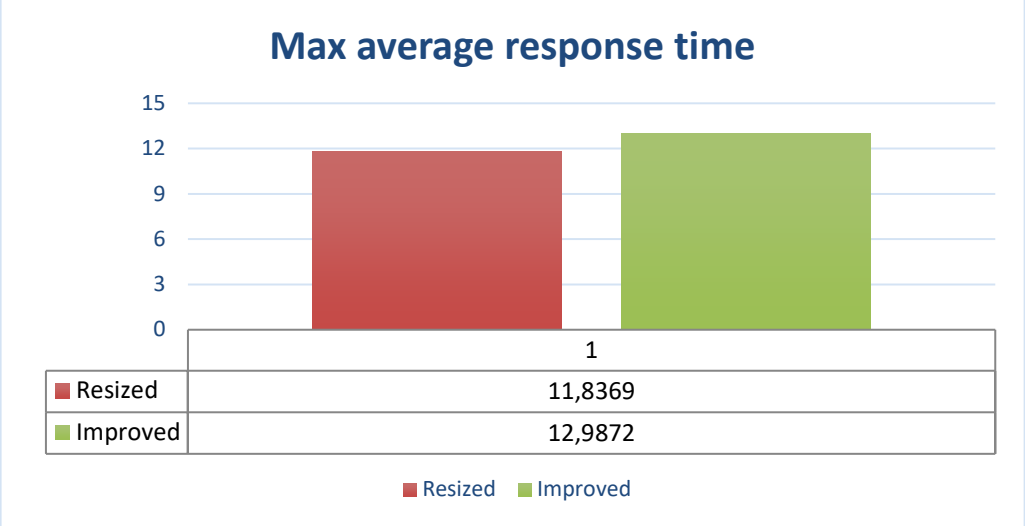
Attraverso le simulazioni effettuate, è stato possibile individuare la percentuale massima di utenti privilegiati rispetto al totale delle utenze per cui il QoS sul massimo tempo di risposta non sia violato ($E(T_S) < 12$):

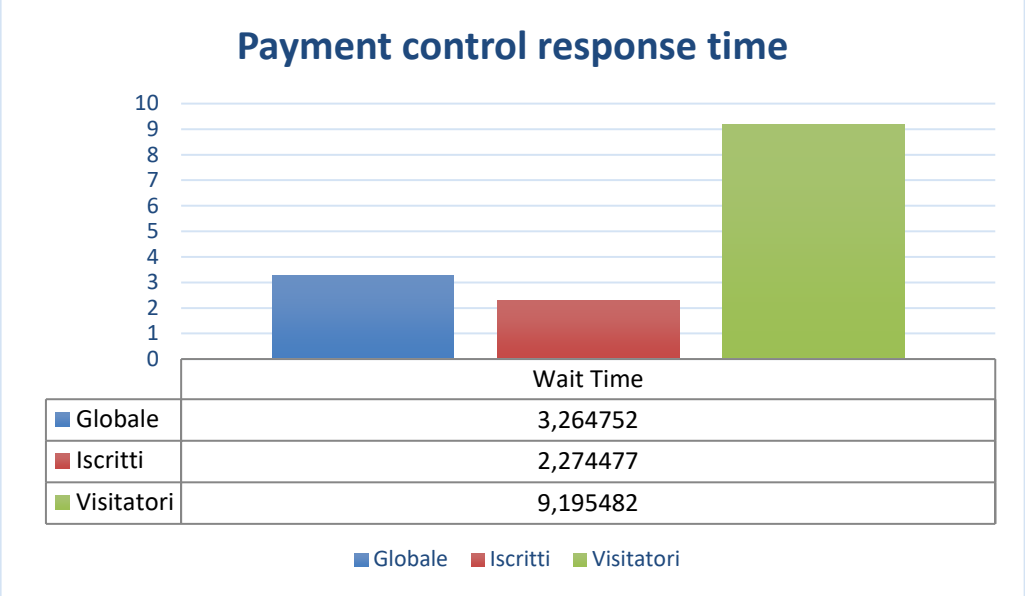
- Tale percentuale risulta essere pari a circa **85.69%**
- Fino a quando gli iscritti non supereranno tale valore rispetto al totale, il QoS sarà rispettato
- Il vantaggio principale è che la probabilità di perdita viene azzerata grazie all'uso di code infinite, garantendo l'assenza di scarto di richieste

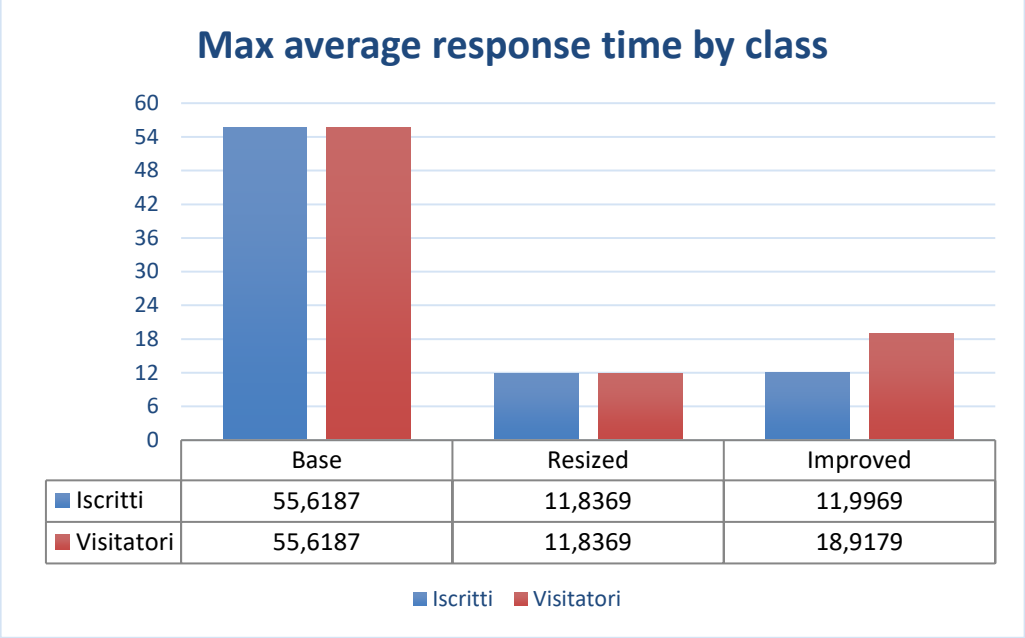
9.4.1 Orizzonte Infinito

Come si evince dai grafici, i tempi peggiorano e il QoS sul massimo tempo di risposta non è più rispettato, tuttavia, utilizzando due code di priorità, è possibile rispettarlo per la maggior parte degli utenti, ovvero quelli iscritti:



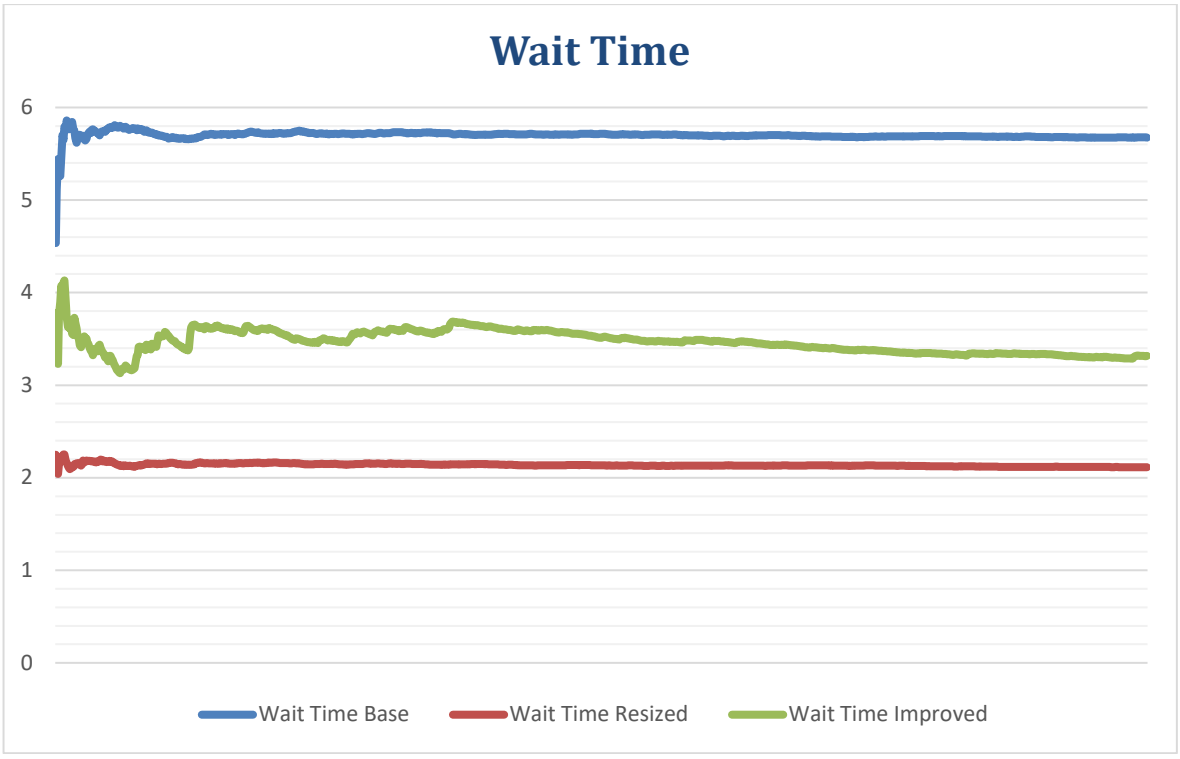


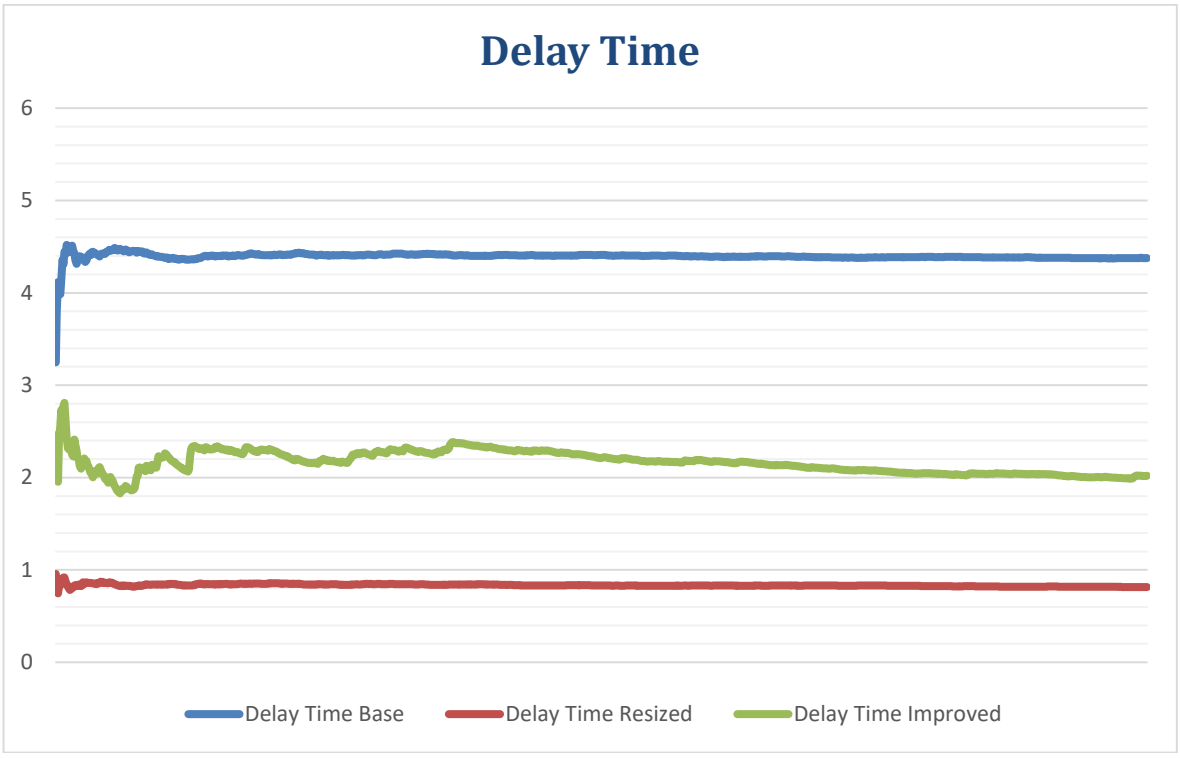

Payment control response time
10
9
8
7
6
5
4
3
2
1
0
Wait Time
Globale 3,264752
Iscritti 2,274477
Visitatori 9,195482
Globale
Iscritti
Visitatori


Max average response time by class
60
54
48
42
36
30
24
18
12
6
0
Base
Resized
Improved
Iscritti 55,6187 11,8369 11,9969
Visitatori 55,6187 11,8369 18,9179
Iscritti
Visitatori

9.4.2 Orizzonte Finito

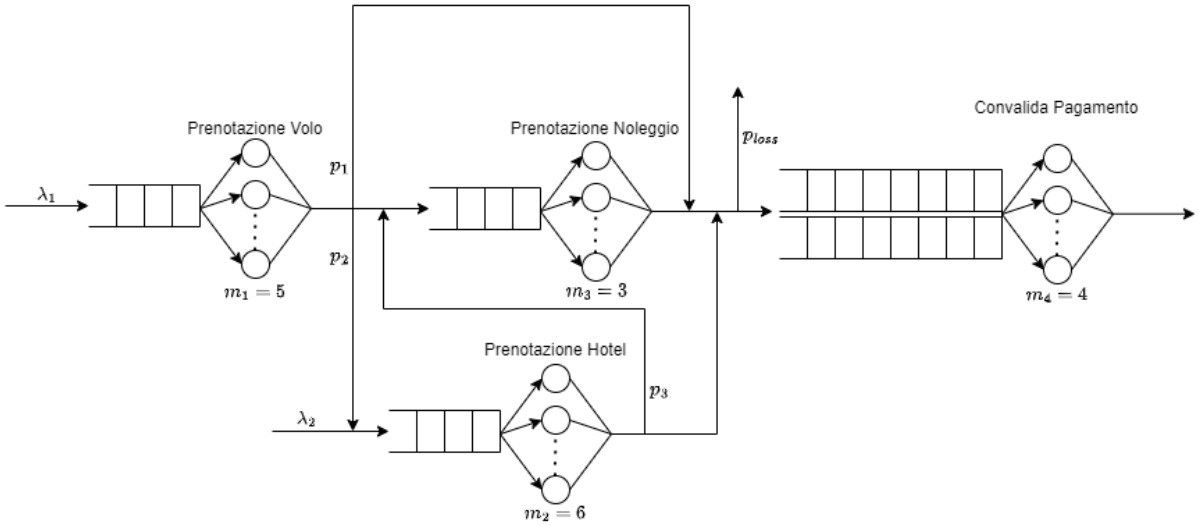
L'unico nodo modificato rispetto alla configurazione di partenza è quello relativo alla convalida dei pagamenti, pertanto si propone un'analisi che si concentra esclusivamente su di esso:





L’aggiunta di una coda infinita garantisce di non rigettare alcuna richiesta e quindi inevitabilmente aumenta l’utilizzazione del nodo, impattando sulla sua stabilità. Infatti, dai grafici è evidente come questa sia estremamente più variabile e più lenta a convergere al valore atteso rispetto alla configurazione base e ridimensionata, che invece erano caratterizzati da una coda limitata.

La nuova topologia dell’infrastruttura per introdurre un ulteriore miglioramento del sistema è la seguente, illustrata in figura:



10 Conclusioni

Per il sistema iniziale e migliorato la configurazione ottima è la seguente:

- $m_1 = 5$
- $m_2 = 6$
- $m_3 = 3$
- $m_4 = 4$

La soluzione migliorativa è una sorta di compromesso tra gli obiettivi prefissati e le richieste dell'azienda, lo scopo è quello di individuare una particolare configurazione che permettesse di eliminare la perdita di richieste e allo stesso tempo soddisfare il QoS sul massimo tempo medio di risposta per gli utenti iscritti:

- Il vantaggio è quello di ottenere un guadagno economico grazie all'assenza di scarto di richieste, garantito dall'introduzione di code infinite.
- Di contro, il primo QoS è rispettato soltanto per l'85% degli utenti (*iscritti*), i restanti 15% (*visitatori*) sperimenteranno tempi di risposta maggiori.

Di seguito, per completezza, sono allegati i risultati prodotti da una fase di verifica e di validazione eseguite sul sistema migliorato dall'introduzione di code infinite.

10.1 Verifica

Consideriamo nuovi controlli di consistenza per quanto riguarda l'ultimo nodo di convalida ed avvalorare la veridicità dei risultati ottenuti. In particolare, possiamo verificare che gli indici di prestazione delle classi siano coerenti con quelli globali:

- p_k = probabilità di appartenere alla classe di priorità *k*
  - $p_1 = 85.69\%$
  - $p_2 = 14.31\%$

Risultati globali

Tempo di interarrivo medio	0.370405 +/- 0.000102
Tempo di risposta medio	3.269364 +/- 0.010376
Tempo di attesa medio	1.969309 +/- 0.010220
Tempo di servizio medio	1.300054 +/- 0.000322
Numero di richieste medio nel nodo	8.827280 +/- 0.028012
Numero di richieste medio nella coda	5.317134 +/- 0.027591
Utilizzazione media	0.877537 +/- 0.000217
Probabilità di perdita	0.00% +/- 0.00%

Risultati classe 1 (Iscritti)

Tempo di interarrivo medio	0.432260 +/- 0.000133
Tempo di risposta medio	2.273157 +/- 0.003061
Tempo di attesa medio	0.973162 +/- 0.002868
Tempo di servizio medio	1.299995 +/- 0.000337
Numero di richieste medio nel nodo	5.259264 +/- 0.007197
Numero di richieste medio nella coda	2.251548 +/- 0.006683
Utilizzazione media	0.751929 +/- 0.000212
Probabilità di perdita	0.00% +/- 0.00%

Risultati classe 2 (Visitatori)

Tempo di interarrivo medio	2.588492 +/- 0.001825
Tempo di risposta medio	9.234981 +/- 0.057328
Tempo di attesa medio	7.934566 +/- 0.057230
Tempo di servizio medio	1.300415 +/- 0.000918
Numero di richieste medio nel nodo	3.568017 +/- 0.022101
Numero di richieste medio nella coda	3.065586 +/- 0.022059
Utilizzazione media	0.125608 +/- 0.000114
Probabilità di perdita	0.00% +/- 0.00%

- Gli arrivi totali devono essere pari alla somma di quelli delle classi distinte
  - $\frac{1}{i_1} + \frac{1}{i_2} = \frac{1}{0.432260} + \frac{1}{2.588492} \approx 2.699748 \approx \frac{1}{0.370405} = \frac{1}{i_{TOT}}$
- La media ponderata del tempo di risposta medio delle classi deve essere pari al tempo di risposta medio globale
  - $P_1 * E(T_{S_1}) + P_2 * E(T_{S_2}) = 0.8569 * 2.273157 + 0.1431 * 9.234981 \approx 3.269394 \approx 3.269364 = E(T_{S_{TOT}})$
- La media ponderata del tempo di attesa medio delle classi deve essere pari al tempo di attesa medio globale
  - $P_1 * E(T_{Q_1}) + P_2 * E(T_{Q_2}) = 0.8569 * 0.973162 + 0.1431 * 7.934566 \approx 1.969339 \approx 1.969309 = E(T_{Q_{TOT}})$
- Il tempo di servizio degli iscritti e dei visitatori dovrebbe essere circa lo stesso
  - $E(S_1) \approx E(S_2) \approx E(S_{TOT}) \leftrightarrow 1.299995 \approx 1.300415 \approx 1.300054 \rightarrow OK$

- Il numero di richieste nel nodo deve essere pari alla somma di quelli delle classi distinte
  - $E(N_{S_1}) + E(N_{S_2}) = 5.259264 + 3.568017 = 8.827281 \approx E(N_{S_{TOT}})$
- Il numero di richieste nella coda deve essere pari alla somma di quelli delle classi distinte
  - $E(N_{Q_1}) + E(N_{Q_2}) = 2.251548 + 3.065586 = 5.317134 \approx E(N_{Q_{TOT}})$
- Il tasso di utilizzazione del nodo deve essere pari alla somma di quelli delle classi distinte
  - $\rho_1 + \rho_2 = 0.751929 + 0.125608 \ = 0.877537 \approx \rho_{TOT}$

10.2 Validazione

Inoltre, possiamo effettuare una validazione su questi valori sulla base di nuove leggi teoriche che sono state derivate durante lo studio eseguito:

- $E(T_{Q_{TOT}}) = P_Q \frac{E(S)}{1-\rho}$
  - $P_Q = \frac{(m\rho)^m}{m!\,(1-\rho)} * P(0)$
  - $P(0) = [\sum_{i=0}^{m-1} \frac{(m\rho)^i}{i!} + \frac{(m\rho)^m}{m!(1-\rho)}]^{-1}$
- $E(T_{S_{CLASS}}) = E(T_{Q_{CLASS}}) + E(S_i)$
- $E(T_{Q_{CLASS}}) = \frac{P_Q E(S)}{\left(1-\sum_{j=1}^{CLASS} \rho_k\right)\left(1-\sum_{j=1}^{CLASS-1} \rho_k\right)}$
  - $E(T_{Q_1}) = \frac{P_Q E(S)}{1-\rho_1}$
  - $E(T_{Q_2}) = \frac{P_Q E(S)}{(1-\rho)(1-\rho_1)}$

A questo punto, abbiamo a disposizione tutte le leggi teoriche necessarie per verificare che effettivamente il modello computazionale è coerente con il modello analitico:

	Modello Analitico	**Modello Sperimentale** ($\boldsymbol{\alpha = 0.01}$)
$E(T_{Q_{TOT}})$	1.97066828289	1.969309 +/- 0.010220
$E(T_{Q_1})$	0.973139119478	0.973162 +/- 0.002868
$E(T_{Q_2})$	7.94399281207	7.934566 +/- 0.057230
$E(T_{S_{TOT}})$	3.27066828289	3.269364 +/- 0.010376
$E(T_{S_1})$	2.27313911948	2.273157 +/- 0.003061
$E(T_{S_2})$	9.24399281207	9.234981 +/- 0.057328

Possiamo concludere che:Chi ha commissionato il lavoro può scegliere quale delle due configurazioni adottare per il suo sistema in base alle esigenze: vuole guadagna mette improved, accetta di perdere dei soldi per scarto di prenotazione per accontentare tutti e dare risposte in tempo breve

Il sistema migliorato è stato verificato e validato, a questo punto l'azienda che ha commissionato il lavoro ha a disposizione due possibili configurazioni per il sistema da poter adottare in base alle esigenze:

- Se l'obiettivo è guadagnare il più possibile, si utilizzerà la configurazione migliorata evitando scarti di richieste, e quindi perdite economiche.
- Altrimenti, si utilizzerà la configurazione ridimensionata garantendo tempi di risposta più brevi, ma accettando di avere perdite economiche per scarti di richieste.

Riferimenti

[1] Hill, R. Discrete-Event Simulation: A First Course. J Simulation 1, 377, 2004.