Università degli Studi di Roma Tor Vergata
Macroarea di Ingegneria
Corso di Laurea Magistrale in Ingegneria Informatica

Performance Modeling of Computer Systems and Networks

A.A. 2024/2025

RELAZIONE PROGETTO

De Orazi Damiano
Matr. 0287069

Mazzucchi Andrea
Matr. 0286349

1 Introduzione

Il progetto si occupa di modellare, simulare e analizzare le prestazioni di un sistema di e-commerce di prodotti alimentari proposto nel testo di Serazzi (par. 6.3 *Simulation of the Workflow of a Web App*).
Il modello in questione nel libro di testo mostra l'utilizzo dei cosiddetti *Class-Switch* in un modello di reti di code classico. Il routing delle richieste è caratterizzato molto spesso da path probabilistici, rendendo difficile la simulazione in alcuni contesti (come in JSIM), ma con i Class-Switch, invece, è possibile descrivere lo stesso routing ma in maniera deterministica. In questo caso, siccome non si utilizza JSIM come ambiente di simulazione, vengono implementati solamente dei Class ID per differenziare le richieste.

2 Architettura di sistema

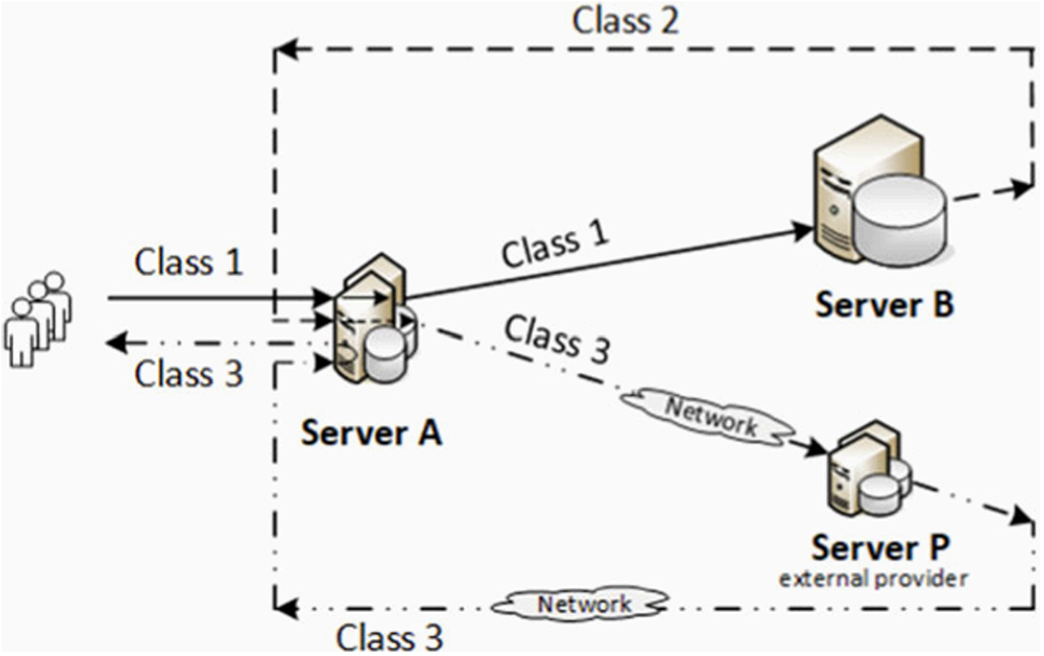


Figura 1

L'architettura del sistema, come mostrato in figura, presenta 3 server:

- **Server A**: è un sistema multicore che si occupa delle esecuzioni di molti servizi lato Front End. Tra questi troviamo: login, processi amministrativi e di CRM, interazione con il servizio di pagamento, operazioni di checkout con update del database dei prodotti, generatore di fatture, servizi di spedizione e tracciamento dei prodotti, e aggiornamento dei dati del cliente.
- **Server B**: è un blade system multiprocessore, altamente scalabile, tollerante ai guasti, con configurazione ridondante per garantire disponibilità continua, fornito di grande quantità di RAM e SSD per i DB. I servizi di cui si occupa riguardano: navigazione del catalogo, processamento del carrello, e gestione dei DB dei prodotti e dei clienti. Inoltre, per garantire il tempo di risposta minimo alle richieste dei clienti, viene implementato un DB in memoria per fare caching degli acquisti più recenti dei clienti.
- **Server P**: è localizzato in un data center di un provider esterno, e si occupa del servizio di pagamento.

Per ridurre la complessità, si sono considerati solo i servizi utili a descrivere il problema e facilitare l'approccio alla soluzione. Questi servizi sono mostrati nella **Figura 2**.

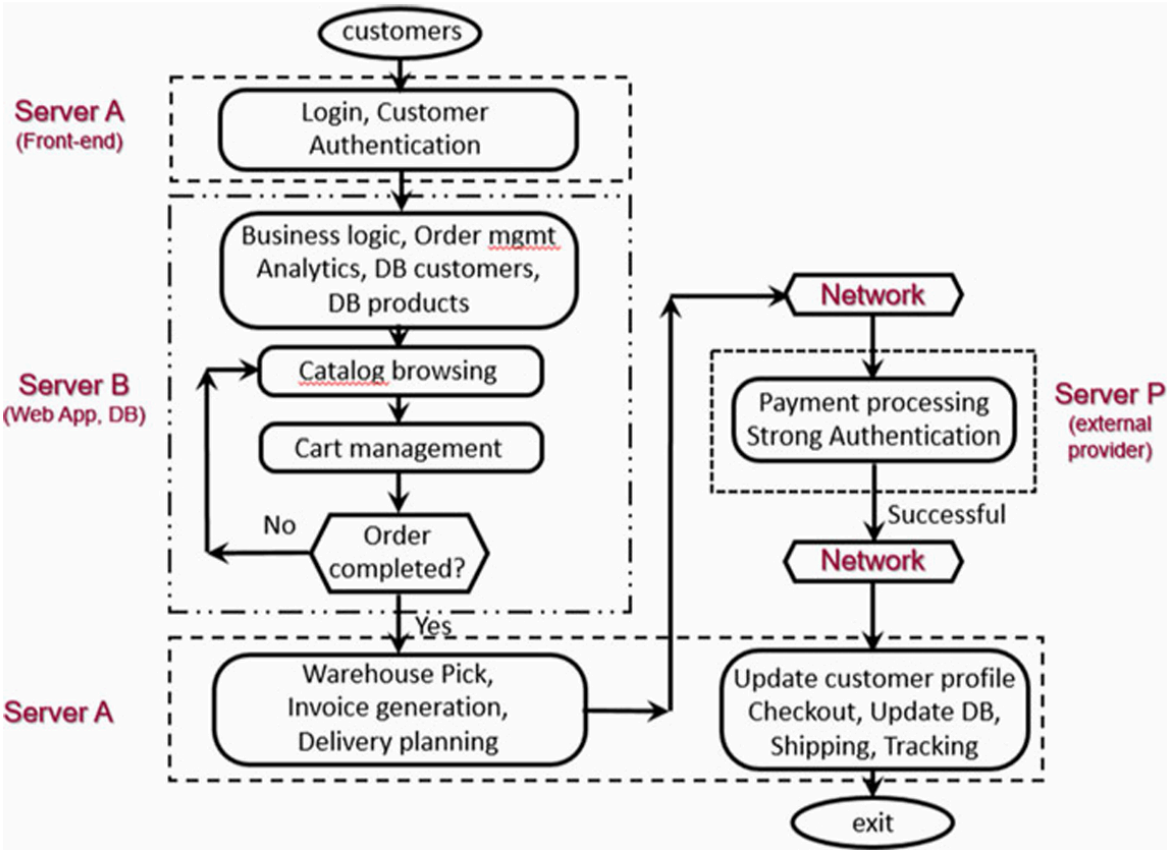


Figura 2

3 Funzionamento del sistema

Secondo la logica di business dell'app e-commerce, l'esecuzione completa di una richiesta richiede tre visite al Server A, una visita al Server B e una visita al Server P. Ad ogni visita al Server A, vengono eseguiti diversi servizi che richiedono tempi medi di servizio diversi. Quindi, la sequenza di visite ai server per considerare una richiesta completata è A-B-A-P-A.

4 Obiettivi

Gli obiettivi preliminari dello studio sono i seguenti:

- determinare le prestazioni dell'app di e-commerce con il livello di autenticazione a un fattore per diversi carichi di lavoro;
- l'impatto sul sistema di un nuovo meccanismo di autenticazione per i pagamenti, l'autenticazione a due fattori;
- il bound al **Throughput** per il sistema corrente nel caso ci sia un aumento del 15% (più precisamente 5000 req/h) rispetto al massimo considerato in precedenza;
- individuazione del bottleneck, con lo studio di una possibile soluzione di scaling orizzontale e/o verticale di uno o più server;
- diversa variabilità degli arrivi e il loro impatto sul sistema.

5 Modello Concettuale

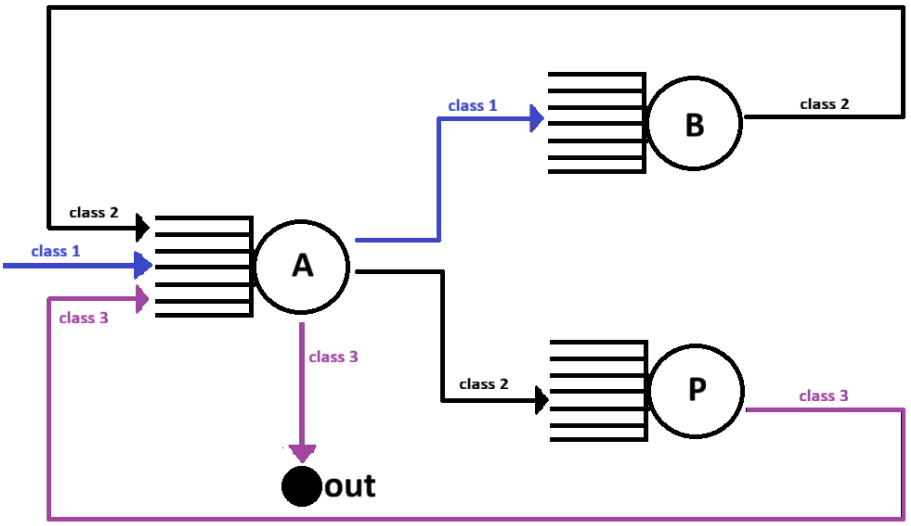


Figura 3

In Figura 3 possiamo vedere la rappresentazione secondo la teoria delle code del sistema come rappresentato in Figura 1, da notare l'assenza dell'elemento Network dei rami che collegano il server P poichè verrà ignorato il delay di trasmissione verso questo server.

I tre centri hanno un solo servente e processano le richieste secondo Processor Sharing (PS). Il routing non è probabilistico in quanto, come già detto, una richiesta una volta entrata nel sistema segue il percorso A-B-A-P-A per poi uscire dal sistema. Quindi non è necessario tener conto di una tabella di routing per questo studio.

Le variabili di stato che caratterizzano il sistema sono:

- il numero di richieste in ogni server, A, B e P al tempo t, nello specifico $\mathbf{x_{A1}(t)}$ il numero di richieste di classe 1 in A, $\mathbf{x_{A2}(t)}$ il numero di richieste di classe 2 in A, $\mathbf{x_{A3}(t)}$ il numero di richieste di classe 3 in A, $\mathbf{x_{B}(t)}$ il numero di richieste di classe 1 in B e $\mathbf{x_{P}(t)}$ il numero di richieste di classe 2 in P.

6 Modello delle specifiche

Le seguenti specifiche del sistema sono fornite e dedotte dal libro di testo:

- arrivi con distribuzione esponenziale, si analizza l'intervallo [0.5, 1,2] con passo 0.05;
- servizi con distribuzione esponenziale, in **Tabella 1**, per l'autenticazione a singolo fattore e **Tabella 2**, per l'autenticazione a 2 fattori.

Stations	Classes		
	1	2	3
Server A (Login, Front end, ...)	0.2	0.4	0.1
Server B (Web App Serv., DBs, ...)	0.8	0	0
Server P (Payment Provider)	0	0.4	0

Tabella 1

Stations	Classes		
	1	2	3
Server A (Login, Front end, ...)	0.2	0.4	0.15
Server B (Web App Serv., DBs, ...)	0.8	0	0
Server P (Payment Provider)	0	0.7	0

Tabella 2

La generazione dei istanti di arrivo e tempi di servizio avverrà tramite un generatore di Lehmer multistream.

7 Modello computazionale

Il programma sviluppato per la simulazione è stato scritto utilizzando Python.

Il programma di partenza da cui si sono derivate le strutture per la gestione del tempo e la generazione del valori è ssq4.py del libro[2].

Per lo sviluppo si è sfruttato il paradigma ad oggetti (OO) per definire diverse classi utili a modellare gli elementi del sistema.

Da questo punto della trattazione le richieste al sistema vengono denominate Job.

Invece di utilizzare il paradigma

Le classi sviluppate o modificate sono le seguenti:

- *Time*, utilizzata per tenere traccia delle scorrere del tempo e degli istanti degli eventi;
- *Track*, tiene traccia del servizio di ogni server;
- *Job*;
- *Server*.

Per la simulazione si è preferito gestire i job in 5 Tipi invece delle 3 classi proposte in precedenza:

- **A1**, i job che arrivano dall'esterno nel Server A;
- **B**, il job risultante dall'esecuzione di A1, processato dal Server B;

- **A2**, il job risultante dall'esecuzione di B, processato dal Server A;
- **P**, il job risultante dall'esecuzione di A2, processato dal Server P;
- **A3**, il job risultante dall'esecuzione di A2, processato dal Server P.

Quindi il cambiamento è stato il seguente:

- Classe 1 -> A1, B;
- Classe 2 -> A2, P;
- Classe 3 -> A3.

La classe Job tiene traccia dell'istante di arrivo, del tempo rimanente di servizio e del tipo.

Gli eventi che avvengono nel corso della simulazione sono così denominati:

1. *arrival_a*, l'evento di arrivo nel sistema;
2. *completion_a*, il completamenti di un Job di tipo A1, A2 o A3;
3. *completion_b*, il completamento di un Job di tipo B,
4. *completion_p*, il completamento di un Job di tipo P.

Non sono presenti altri eventi di arrivo, ad esempio l'arrivo di un job di tipo P sul Server P, perché tutti questi eventi sono istantanei dopo il rispettivo completamento, A2 genera P e l'arrivo viene subito processato.

Per quanto riguarda l'implementazione di PS lo schema è il seguente:

- ➔ **arrivo su server**: vengono aggiornati tutti i tempi di servizio rimanenti dei job attualmente nel server e dopo aver aggiunto il nuovo job viene calcolato il tempo del successivo completamento
- ➔ **completamento su server**: vengono aggiornati tutti i tempi di servizio dei job presenti, rimosso il job che ha completato l'esecuzione, generato se necessario un nuovo job per un altro server e calcolato il nuovo tempo di completamento

Per ovviare alle imprecisioni legate al calcolo floating point per rimuovere il job dalla lista questa viene riordinata per tempo rimanente di esecuzione e rimosso il primo job. Si è verificato che l'imprecisione dovuta al calcolo floating point in questo caso è dell'ordine 1×10^{-11}.

```python
class Server:
    def __init__(self):
        self.jobs = []
        self.number = 0  # number in the node
        self.index = 0  # used to count departed jobs
        self.area = Track()
        self.last_event = 0

        # interarrival
        self.arrivals = 0
        self.last_arrival = 0
        self.avg_interarrival = 0
        self.interarrival_variance = 0

        # service
        self.avg_service = 0
        self.service_variance = 0

    def get_min_remaining_process_time(self):...

    def get_next_complete_process_time(self):...

    def reset_stats(self, current_time):...

    def process_arrival(self, new_job):...

    def process_completion(self, completion_time):...
```

Codice 1

In **Codice 1** è possibile vedere la classe Server con i suoi metodi ed attributi.

Per il calcolo del tempo medio di interarrivo e di servizio viene utilizzato l'algoritmo di Welford, infatti tra gli attributi sono presenti sia la media che la varianza.

Il metodo *process_arrival* permette di aggiornare il tempo rimanente di servizio dei job presenti, gestire l'aggiornamento delle statistiche di interarrivo e aggiungere il nuovo job.

Il metodo *get_next_complete_process_time* restituisce il tempo necessario affinché il job, con tempo di servizio rimanente minore completi l'esecuzione.

Il metodo *process_completion* permette di aggiornare il tempo rimanente di servizio dei job presenti, rimuovere il job che completa l'esecuzione e aggiornare le statistiche relative al tempo di servizio.

La funzione di simulazione consta di un ciclo while che permette da sistema vuoto di gestire tutti gli eventi in ordine cronologico, fermare gli arrivi ad un tempo desiderato e riportare il sistema allo

stato iniziale. Al termine di simulazione si ottengono tutte le statistiche dei singoli server e di quelle globali.

8 Verifica

Per l'operazione di verifica, noto che il sistema è stabile, si sono calcolati i flussi in entrata ai singoli server e le prestazioni dei Server B e P secondo i modelli analitici per poi confrontarli con i risultati della simulazione

Per quanto riguarda l'analisi dei flussi si ha λ come flusso in entrata dall'esterno del sistema, λ_A flusso totale in A, λ_B e λ_P flussi rispettivamente in B e P. Di seguito viene illustrato il sistema del bilanciamento dei flussi:

$$\lambda_A = \lambda + \lambda_B + \lambda_P$$
$$\lambda_B = \lambda$$
$$\lambda_P = \lambda$$

Per verificare i risultati teorici del sistema del bilanciamento dei flussi nel caso steady-state, si utilizza la tecnica delle batch means con 64 batch e 8192 campioni; il campionamento avviene in base agli arrivi dall'esterno nel sistema, quindi le richieste di tipo A1 (verrà utilizzato lo stesso campionamento nei successivi usi della tecnica di batch).

λ	Server	Risultato Analitico	Risultato Simulazione
0.5	A	0.66666666666666	0.667092458395113 ± 0.0008396755540814445
0.5	B	2	2.001217632360312 ± 0.0025233660431006927
0.5	P	2	2.0012188625266116 ± 0.002518706324430831
1.2	A	0.27777777777777	0.27795759665723746 ± 0.000335557864306096
1.2	B	0.83333333333333	0.8338391574588223 ± 0.0010519363718811104
1.2	P	0.83333333333333	0.833851159216368 ± 0.0010011214651201572

Tabella 3

$$E(N_S)^{M/G/1/PS} = \frac{\rho}{1-\rho} = E(N_S)^{M/M/1/FIFO}$$
$$E(T_S)^{M/G/1/PS} = \frac{E(S)}{1-\rho} = E(T_S)^{M/M/1/FIFO}$$

Per verificare il funzionamento della classe Server si è opportunamente simulato un singolo server così da verificare i risultati teorici nel caso steady-state, utilizzando la tecnica delle batch means con 64 batch e 8192 campioni.

Vengono riportati ad esempio i risultati per:

- λ = 0.5 e E[S] = 0.7, per un carico basso;
- λ = 1.2 e E[S] = 0.7, per un carico alto;

λ = 0.5 e E[S] = 0.7

Metrica	Risultato Analitico	Risultato Simulazione
Interarrivo medio	2	2.0018001290615186 ± 0.005528279677442084
Tempo medio di servizio	1.076923077	1.0756891284500347 ± 0.006620739138871457
Popolazione Media	0.5384615385	0.5374836708571215 ± 0.004018654433565858
Utilizzazione	0.35	0.349786586021274 ± 0.0015382751322791225

Tabella 4

λ = 1.2 e E[S] = 0.7

Metrica	Risultato Analitico	Risultato Simulazione
Interarrivo medio	0.833333333	0.8340833871089666 ± 0.0023034498656009884
Tempo medio di servizio	4.375	4.357594985806601 ± 0.1457908428013647
Popolazione media	5.25	5.233034196882465 ± 0.18253638398673142
Utilizzazione	0.84	0.8394898009658016 ± 0.0037220564458447527

Tabella 5

Dati i risultati possiamo affermare che l'implementazione del server PS è corretta.

Mancando un risultato analitico non è possibile fare un verifica per il Server A poiché gestisce dei job con diversa distribuzione dei tempi di servizio.

9 Validazione

Per la validazione del sistema si sono presi come riferimento i risultati del testo[1] per il sistema “di base”, ovvero con autenticazione a 1 fattore.

Metrica	Risultato	Simulazione
Popolazione Media (λ = 0.5)	1.4	1.4550211536613253 ± 0.003859999801210321
Popolazione Media (λ = 1.2)	30.81	29.882047877961142 ± 1.4307264944893932
Tempo medio di servizio (λ = 0.5)	2.9	2.911705442634646 ± 0.005550582129043626
Tempo medio di servizio (λ = 1.2)	25.8	24.86651435020122 ± 1.174634334605448

Tabella 6

Dai risultati mostrati dalle Tabelle 5 si evince che il modello implementato rispetta i valori del modello del testo.

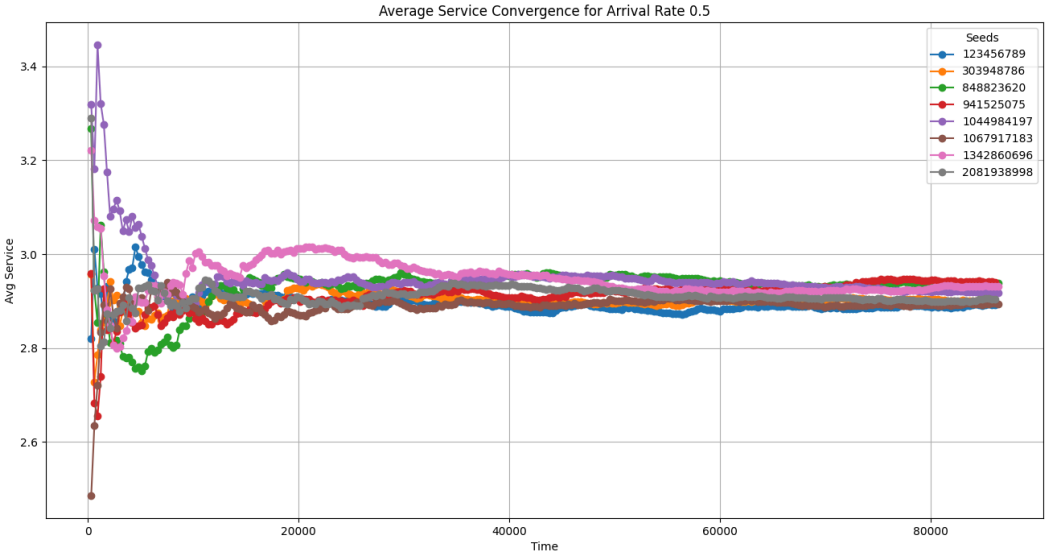
10 Autenticazione a un Fattore

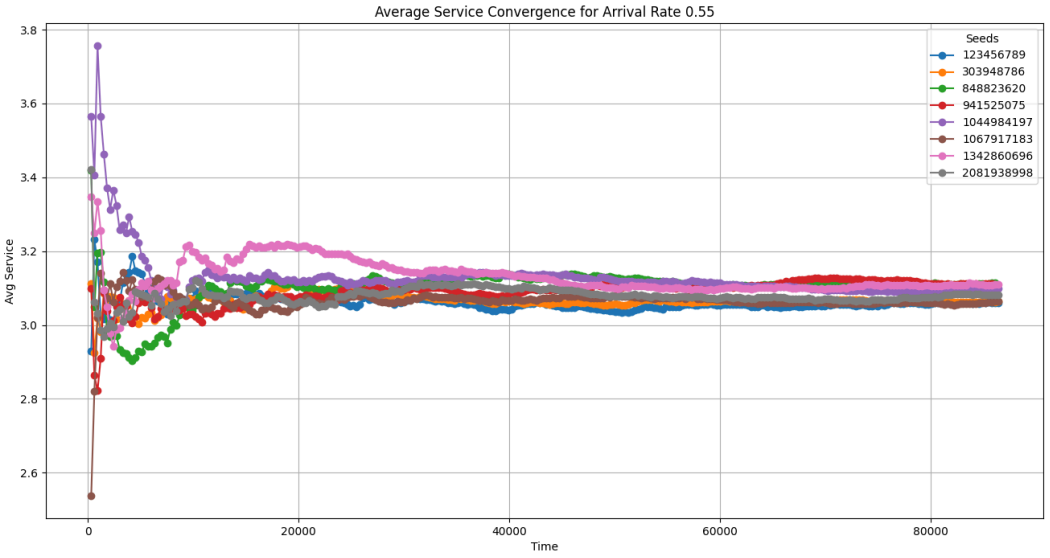
L’analisi del sistema inizia con il prevede l'analisi del transitorio, ovvero la verifica di convergenza a valori stazionari nel breve periodo della metrica di riferimento, in questo caso il tempo di servizio.

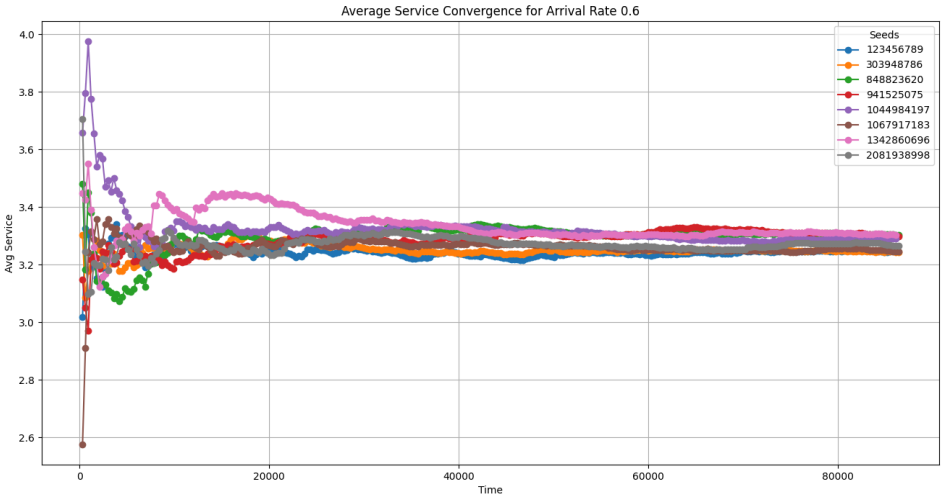
10.1 Analisi transitorio

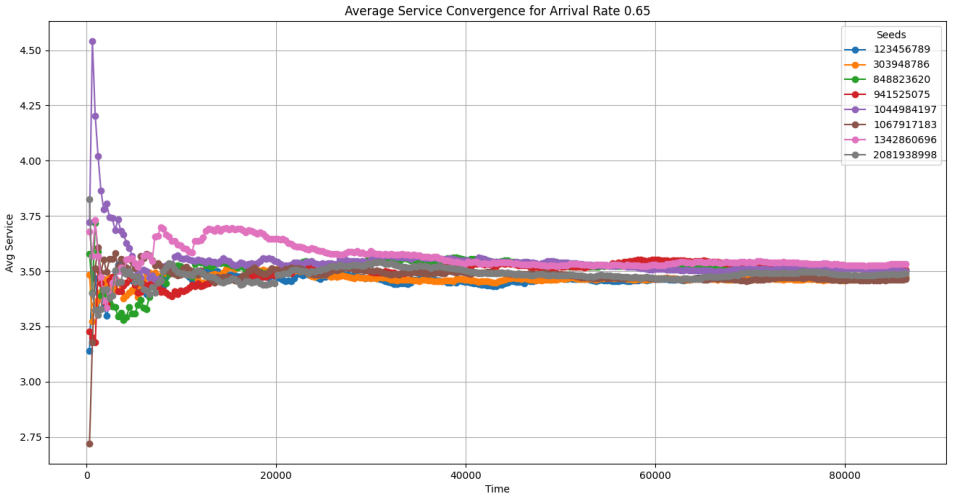
L’analisi del transitorio è stata svolta secondo le seguenti condizioni:

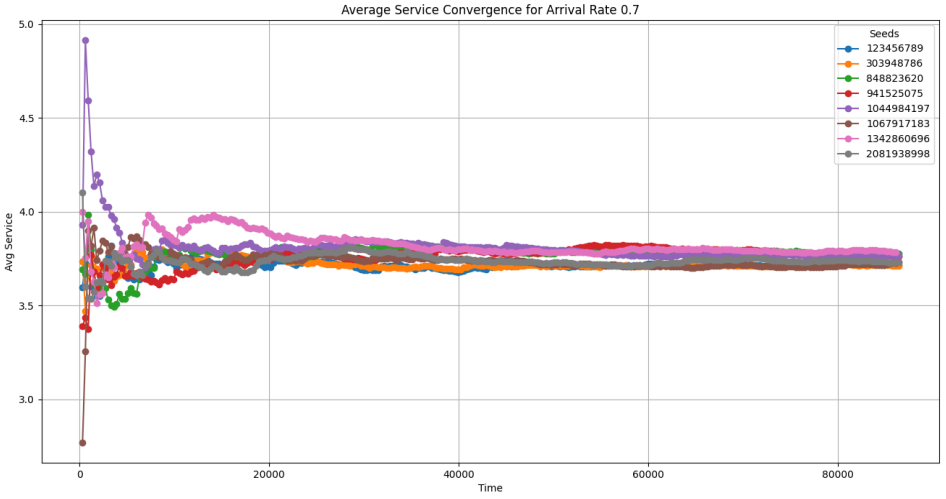
- durata del periodo: 24 ore
- flussi di entrata analizzati: 0.5 j/s e 1.2 j/s
- condizione iniziale: sistema vuoto

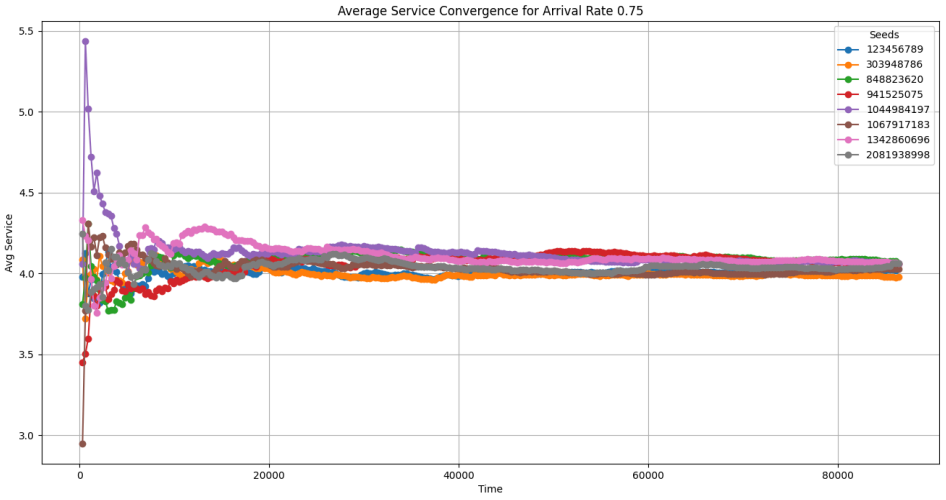

Average Service Convergence for Arrival Rate 0.5
Seeds
123456789
303948786
848823620
941525075
1044984197
1067917183
1342860696
2081938998
Avg Service
Time
2.6
2.8
3.0
3.2
3.4
0
20000
40000
60000
80000

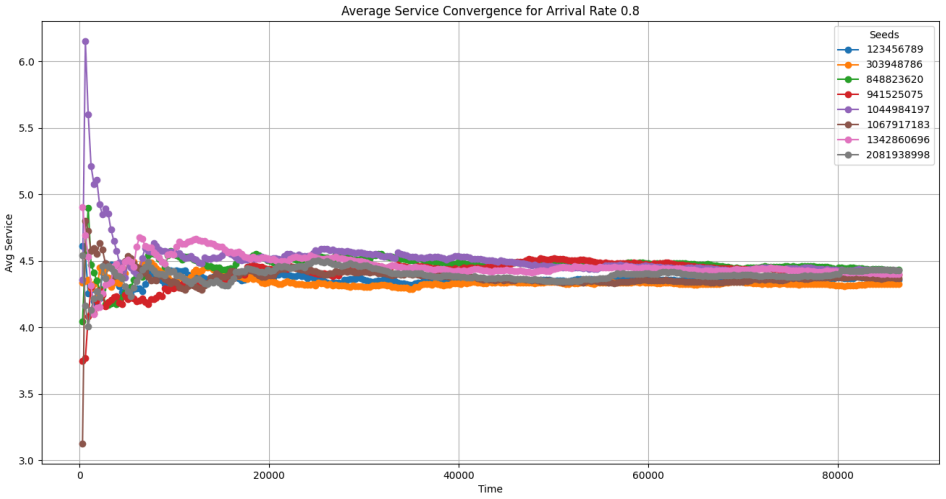

Average Service Convergence for Arrival Rate 0.55
Seeds
123456789
303948786
848823620
941525075
1044984197
1067917183
1342860696
2081938998
Avg Service
Time
2.6
2.8
3.0
3.2
3.4
3.6
3.8
0
20000
40000
60000
80000

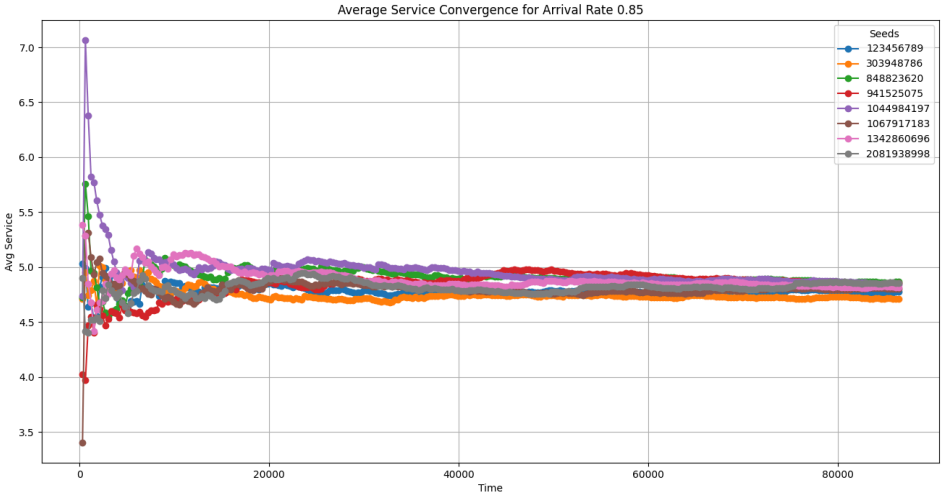

Average Service Convergence for Arrival Rate 0.6
Avg Service
Time
Seeds
123456789
303948786
848823620
941525075
1044984197
1067917183
1342860696
2081938998

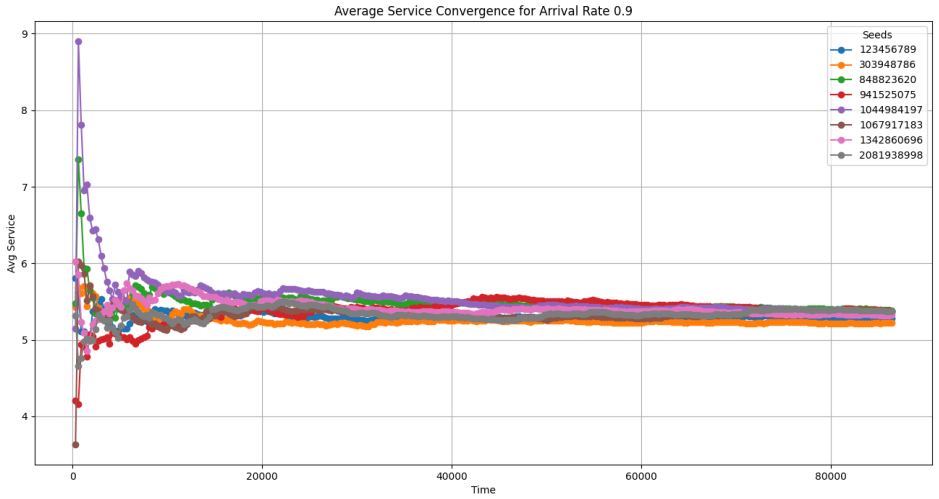

Average Service Convergence for Arrival Rate 0.65
Avg Service
Time
Seeds
123456789
303948786
848823620
941525075
1044984197
1067917183
1342860696
2081938998

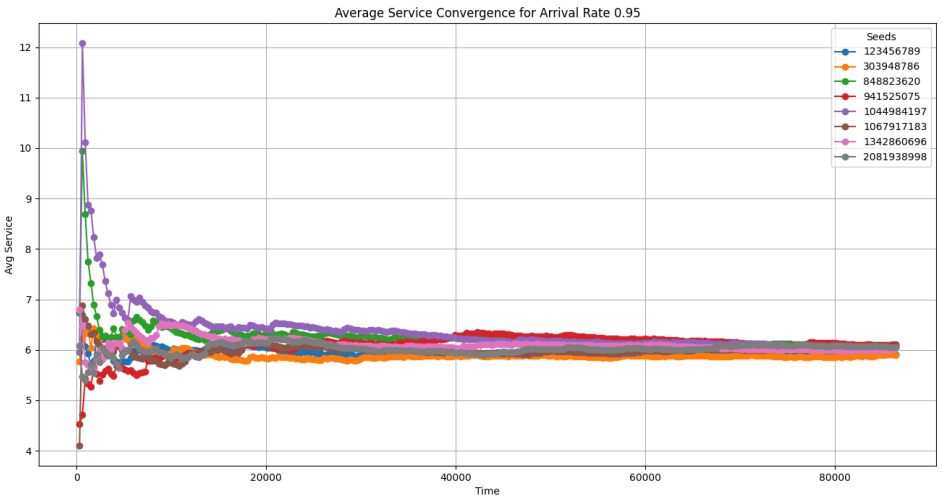

Average Service Convergence for Arrival Rate 0.7
Avg Service
Time
Seeds
123456789
303948786
848823620
941525075
1044984197
1067917183
1342860696
2081938998

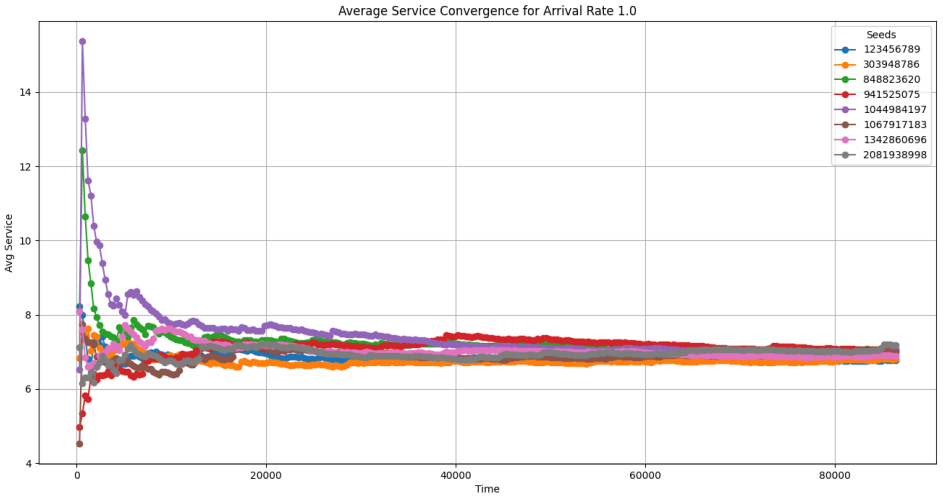

Average Service Convergence for Arrival Rate 0.75
Avg Service
Time
Seeds
123456789
303948786
848823620
941525075
1044984197
1067917183
1342860696
2081938998

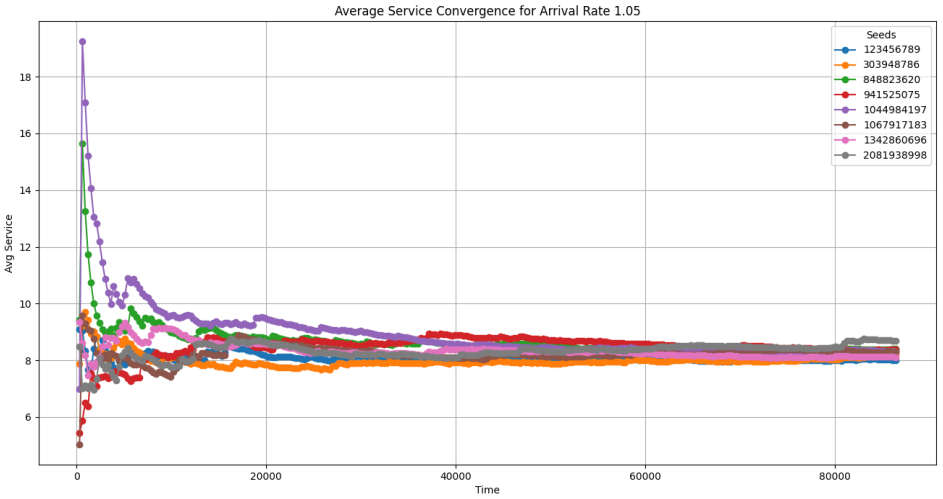

Average Service Convergence for Arrival Rate 0.8
Avg Service
Time
Seeds
123456789
303948786
848823620
941525075
1044984197
1067917183
1342860696
2081938998

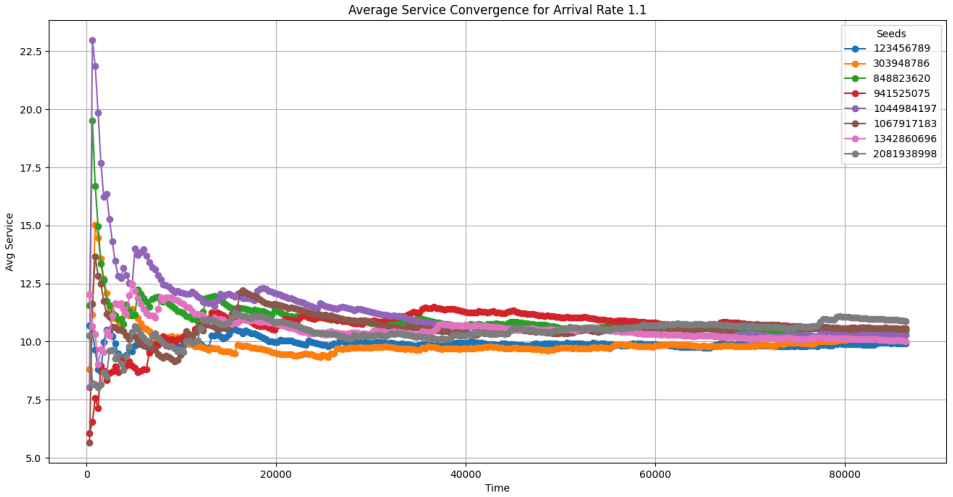

Average Service Convergence for Arrival Rate 0.85
Avg Service
Time
Seeds
123456789
303948786
848823620
941525075
1044984197
1067917183
1342860696
2081938998

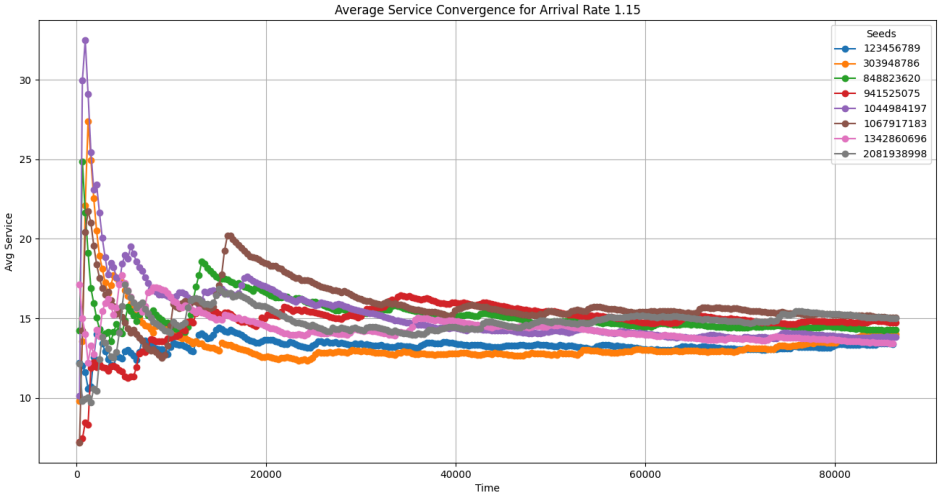

Average Service Convergence for Arrival Rate 0.9
Avg Service
Time
Seeds
123456789
303948786
848823620
941525075
1044984197
1067917183
1342860696
2081938998

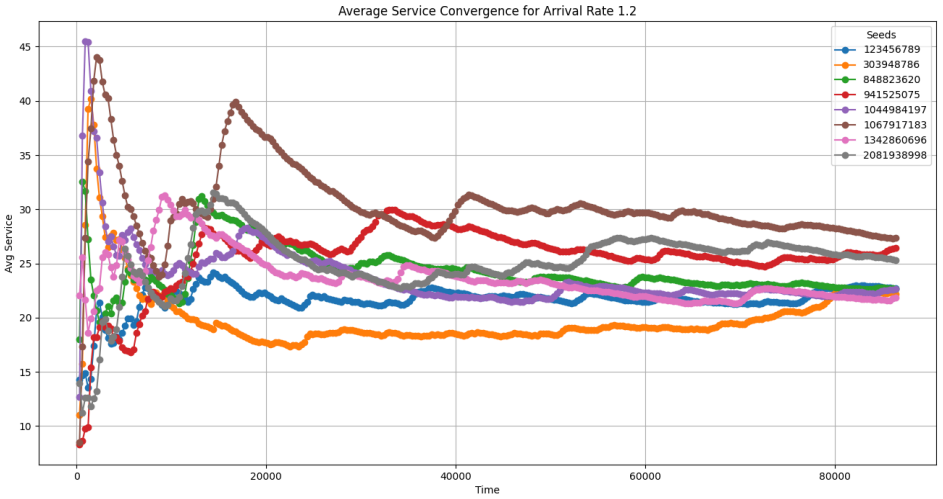

Average Service Convergence for Arrival Rate 0.95
Avg Service
Time
Seeds
123456789
303948786
848823620
941525075
1044984197
1067917183
1342860696
2081938998


Average Service Convergence for Arrival Rate 1.0
Avg Service
Time
Seeds
123456789
303948786
848823620
941525075
1044984197
1067917183
1342860696
2081938998


Average Service Convergence for Arrival Rate 1.05
Avg Service
Time
Seeds
123456789
303948786
848823620
941525075
1044984197
1067917183
1342860696
2081938998


Average Service Convergence for Arrival Rate 1.1
Avg Service
Time
Seeds
123456789
303948786
848823620
941525075
1044984197
1067917183
1342860696
2081938998


Average Service Convergence for Arrival Rate 1.15
Avg Service
Time
Seeds
123456789
303948786
848823620
941525075
1044984197
1067917183
1342860696
2081938998



In tutti i grafici si nota che il sistema converge in un tempo che varia tra i 40000 e 60000 secondi, eccetto per l'ultimo caso, ossia per λ=1.2, il quale ha bisogno di un tempo maggiore per raggiungere la convergenza.

10.2 Analisi del transiente

Per quanto riguarda lo studio dei valori stazionari d'interesse in questo caso di studio includiamo il tempo medio di risposta e l'utilizzazione poiché verrà utilizzata per l'individuazione dei bottleneck.

Vengono i riportati i valori limiti dei tassi di arrivo come nei casi precedenti

Metrica	λ	Simulazione
E(S)	0.5	2.911705442634646 ± 0.005550582129043626
ρ_A	0.5	0.34963713612684294 ± 0.0006267818344796862
ρ_B	0.5	0.4001423157235962 ± 0.0006857304103566013
ρ_P	0.5	0.19974162233006676 ± 0.00034207117076511076
E(S)	1.2	24.86651435020122 ± 1.174634334605448
ρ_A	1.2	0.8391186785159244 ± 0.0014636041699630216
ρ_B	1.2	0.9603298727790543 ± 0.0015948193899785281
ρ_P	1.2	0.4793737216434306 ± 0.0007747292144484478

Tabella 7

11 Esperimenti

Nei paragrafi seguenti verranno sperimentate diverse modifiche del sistema con diversi gradi di complessità.

Data la mancanza di informazioni utili ad identificare un periodo di tempo significativo per effettuare studi del transitorio, relativi ad esempio a fasce orarie con picchi di arrivo, si è posta l'attenzione all'analisi del transiente tramite batch means.

11.1 Autenticazione a due Fattori

L'introduzione dell'autenticazione a 2 fattori prevede l'aumento del tempo di servizio dei job di tipo A3 e P come si deduce dalla **Tabella 2**.

Metrica	λ	Simulazione
E(S)	0.5	3.6109656461846615 ± 0.0067097347364646585
ρ_A	0.5	0.37466846806471804 ± 0.0006314696940886895
ρ_B	0.5	0.40017012278309894 ± 0.0007496798728471607
ρ_P	0.5	0.3495731233388057 ± 0.0006822915722964509
E(S)	1.2	31.51485062891929 ± 1.281192425734563
ρ_A	1.2	0.8991903854173976 ± 0.0014847765532154768
ρ_B	1.2	0.9604044809388099 ± 0.0017366668918975417
ρ_P	1.2	0.8389689466431051 ± 0.0015888454760994957

Tabella 8

Confrontando i valori di **Tabella 7**, è evidente l'impatto della modifica sui tempi di servizio globali e sull'utilizzazione dei due server coinvolti, A e P.

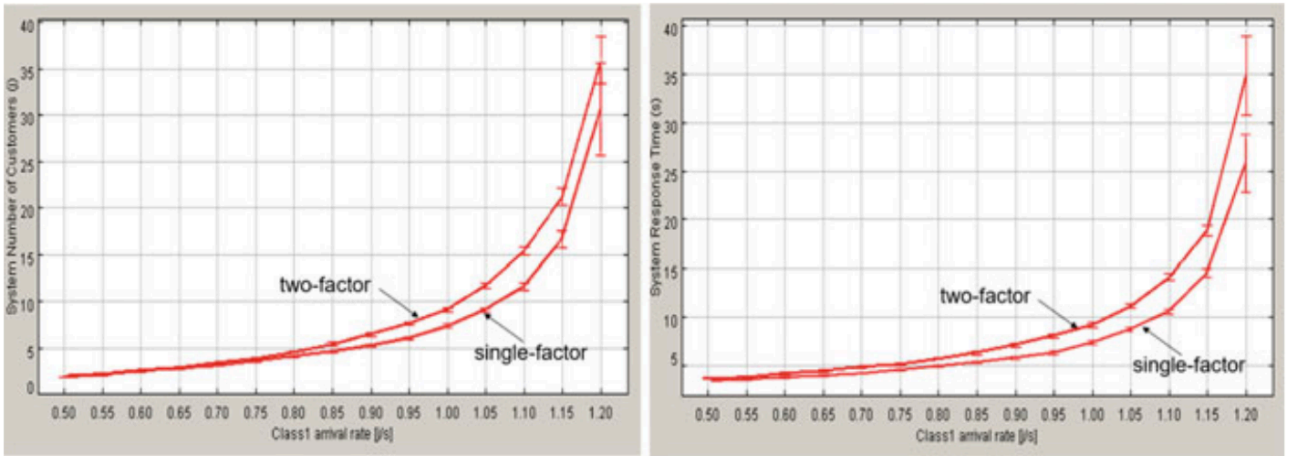


Figura 4

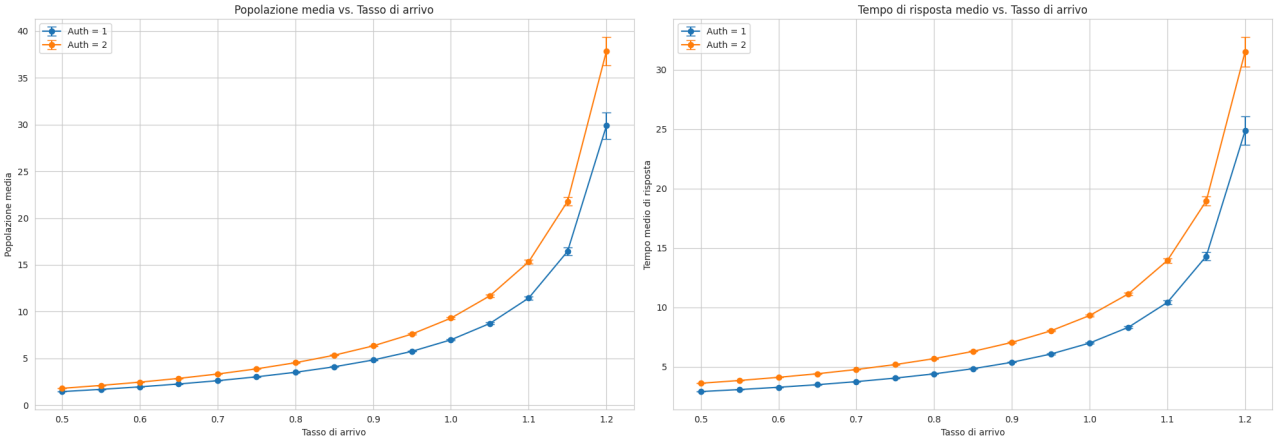


Grafico 1

Dal **Grafico 1** è possibile apprezzare l'aumento della popolazione media e del tempo medio di risposta dovuto all'autenticazione a due fattori.

Nel confronto con il grafico del testo[1] per quanto riguarda il sistema a due fattori si ha una differenza nel tempo medio di risposta di circa 3 s.

11.2 Bottleneck e analisi del Throughput

L'individuazione del bottleneck è chiara: il server B, da **Tabella 7** sia con un carico basso che alto ha l'utilizzazione maggiore nel sistema. Il bound al Throughput è dato da B, che lo pone a 1.25 req/s ($1/D_{max}$ = 1.25 req/s, dove D_{max} = 0.8 s).

11.2.1 Scaling Verticale Server B

Come suggerito dal testo[1] il server B verrà scalato verticalmente, quindi il job B avrà un tempo di servizio medio di 0.4 s invece di 0.8 s. Il throughput a questo punto è limitato dal server A, essendo il nuovo bottleneck. Essendo ora D_{max} = 0.7 s, ovvero la somma delle domande dei job di A si ha il nuovo bound al throughput a 1.4 req/s.

Metrica	λ	Simulazione
E(S)	0.5	2.0739440651513044 ± 0.005941054990885395
ρ_A	0.5	0.3493547556559323 ± 0.0012414064619625624
ρ_B	0.5	0.199766228188276 ± 0.0008210538433733723
ρ_P	0.5	0.19942574425073245 ± 0.0007227366508650686
E(S)	1.2	5.852818517402003 ± 0.10840443011028844
ρ_A	0.5	0.8384457577838306 ± 0.0029727144675486014
ρ_B	1.2	0.47943923054693643 ± 0.0019737530034143465
ρ_P	1.2	0.4786230042638306 ± 0.001757977258467188
E(S)	1.4	31.46022831662848 ± 5.192670348693403

ρ_A	0.5	0.978153512941929 ± 0.0034013439858984877
ρ_B	1.4	0.5593349505981215 ± 0.002279914990119292
ρ_P	1.4	0.5583875338713618 ± 0.002038658754283485

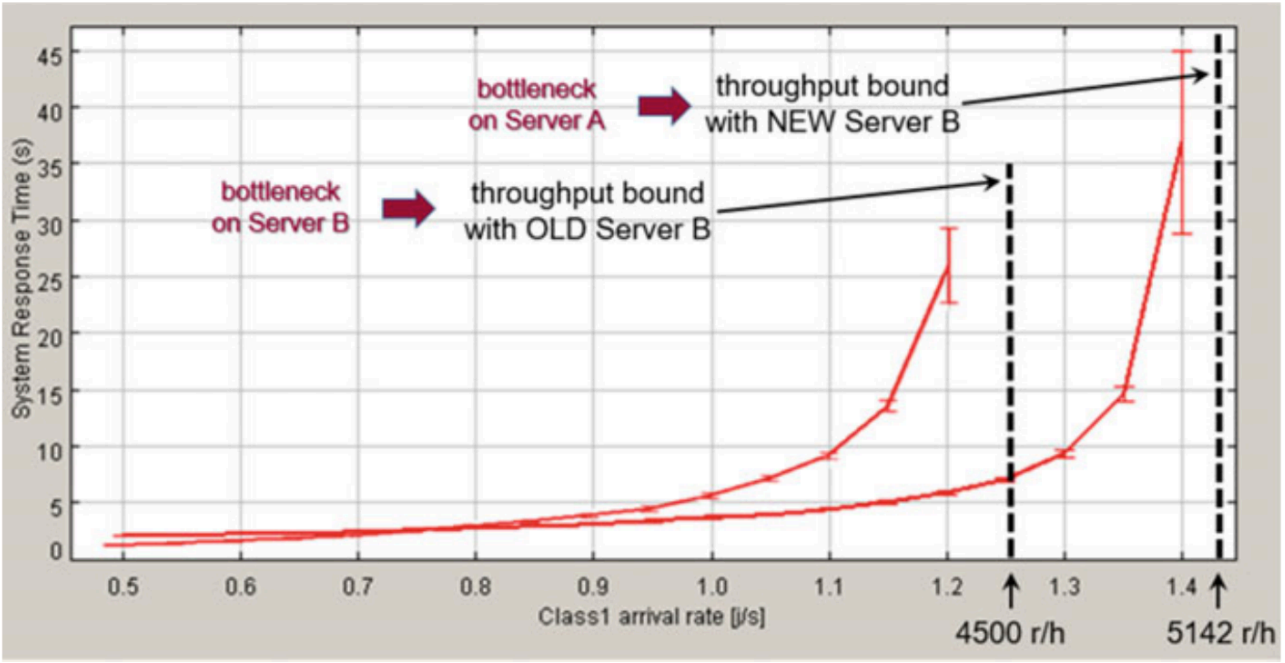


Figura 5

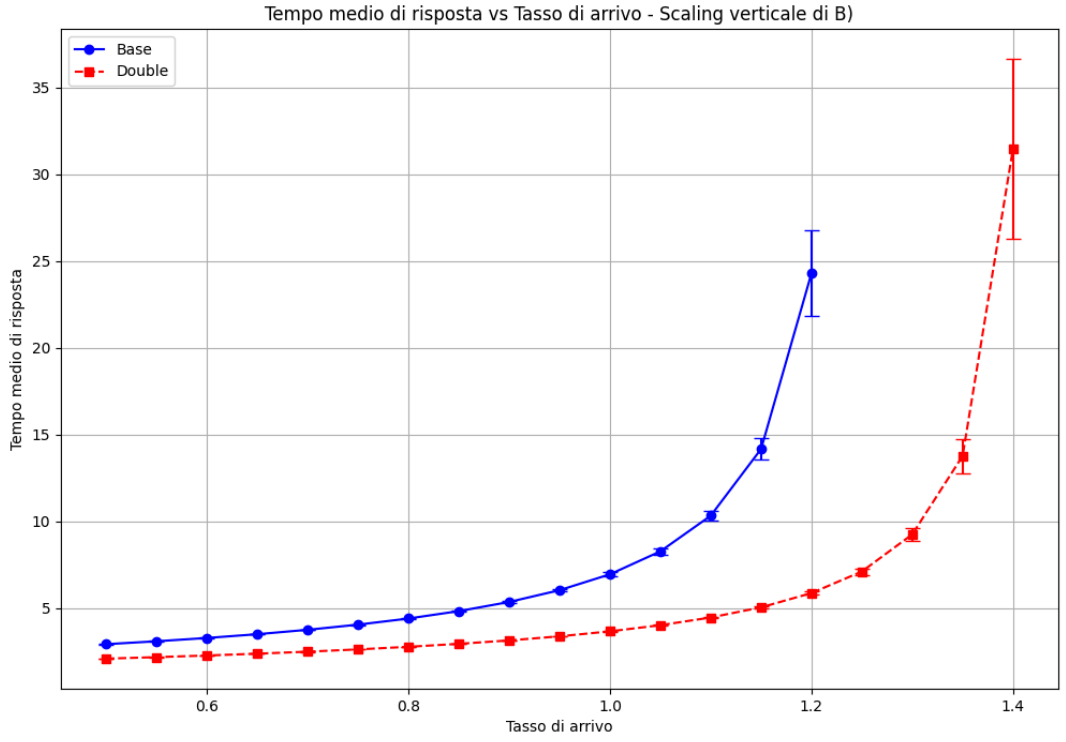


Grafico 2

11.2.2 Scaling Orizzontale Server A

Dato il nuovo bound al Throughput per migliorare le prestazioni si è pensato di effettuare uno scaling orizzontale del server con load balancing round robin, modellato con una bernoulliana di parametro 0.5.

Metrica	λ	Simulazione
E(S)	0.5	1.846512926813716 ± 0.0036537987810935325
ρ_{A1}	0.5	0.17469979560057544 ± 0.0007644986194562248
ρ_{A2}	0.5	0.1746572937226281 ± 0.0007217940639138821
ρ_{B}	0.5	0.199765476539416 ± 0.0008209956523783382
ρ_{P}	0.5	0.19942609084243285 ± 0.0007229220445315387
E(S)	1.2	2.7372260213961996 ± 0.011939694676602585
ρ_{A1}	0.5	0.4186485777633767 ± 0.0016736967057390955
ρ_{A2}	0.5	0.4198095554628995 ± 0.0020279176995589374
ρ_{B}	1.2	0.47943736801040393 ± 0.0019714570475531834
ρ_{P}	1.2	0.4786252881263622 ± 0.001750798895985887
E(S)	1.4	3.1827729092666783 ± 0.019810743769482247
ρ_{A1}	0.5	0.4897103023035319 ± 0.0021165545380634836
ρ_{A2}	0.5	0.4884892556597438 ± 0.002295786918765311
ρ_{B}	1.4	0.5593487903581241 ± 0.002298724248757984
ρ_{P}	1.4	0.5584006696828504 ± 0.002043377626962324

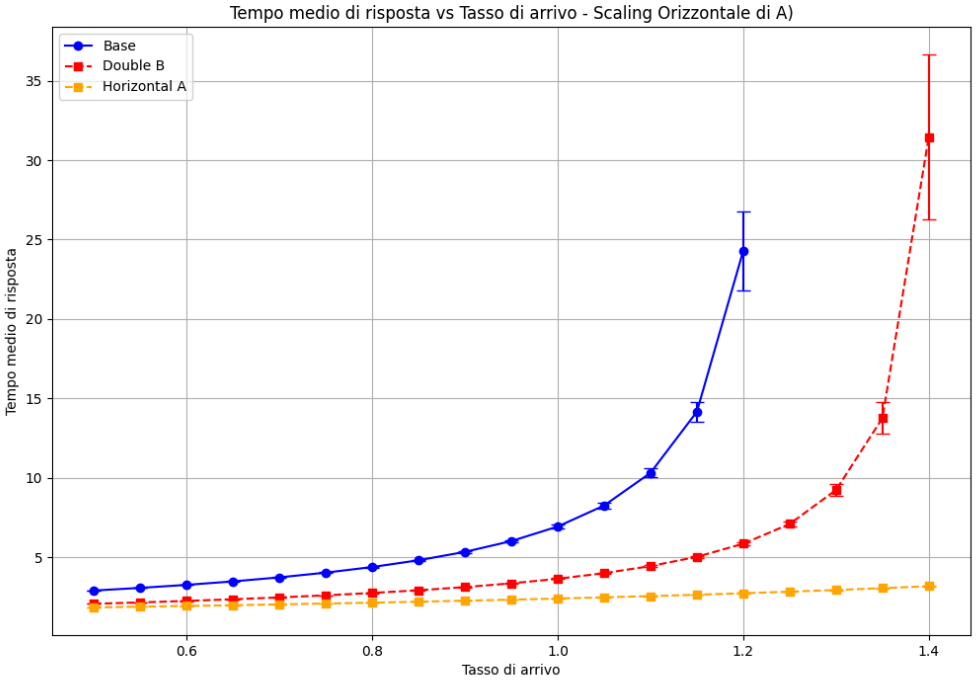


Grafico 3

Dopo questo scaling è di nuovo il Server B insieme al Server P a determinare con una $D_{max} = 0.4$ in il nuovo bound al Throughput di 2.5 req/s. Date le specifiche del sistema migliorare ulteriormente il server B sarebbe pressoché inutile poichè il server P, gestito da un provider esterno, determinerebbe comunque lo stesso bound.

11.3 Flusso in entrata iperesponenziale

Una diversa configurazione del sistema suggerita dal testo[1] prevede un flusso in entrata caratterizzato da una distribuzione iperesponenziale. L'esperimento prevede di analizzare il sistema con autenticazione a un fattore nell'intervallo dei tassi di arrivo [0.5,1.2] nei casi p, probabilità degli stati, 0.1 e 0.2.

Metrica	λ	p	Simulazione
E(S)	0.5	0.1	3.9140226236098226 ± 0.022264391099912288
ρ_A	0.5	0.1	0.350304085946673 ± 0.00250932223258622
ρ_B	0.5	0.1	0.39956497400767443 ± 0.002713829693301275
ρ_P	0.5	0.1	0.20012303875704476 ± 0.001497596613922472
E(S)	0.5	0.2	3.3202950881491264 ± 0.01415884088939848
ρ_A	0.5	0.2	0.34915782716138233 ± 0.0016393515440271524
ρ_B	0.5	0.2	0.39827061576948836 ± 0.0018703565632641715

Metrica	λ	p	Simulazione
ρ_P	0.5	0.2	0.19946818584050383 ± 0.0010189670294319777
E(S)	1.2	0.1	52.52226443824536 ± 6.824111207720179
ρ_A	1.2	0.1	0.8406107641619277 ± 0.005764251317573327
ρ_B	1.2	0.1	0.8330138837149186 ± 0.005715330242238054
ρ_P	1.2	0.1	0.48021656296520054 ± 0.0033841944953426875
E(S)	1.2	0.2	32.19643624428459 ± 3.325986890954446
ρ_A	1.2	0.2	0.8378867065314501 ± 0.0038656093558737067
ρ_B	1.2	0.2	0.9557663662874352 ± 0.0042141643415092465
ρ_P	1.2	0.2	0.4786672162833436 ± 0.00237803475994608

Tabella 9

Confrontando i valori della **Tabella 9** con quelli della **Tabella 7** relativa all'analisi del transiente del sistema ad un fattore, si nota che i tempi di risposta relativi al tasso λ=0.5 sono aumentati sia per p=0.1 che per p=0.2, ma l'aumento più significativo viene registrato dal tempo di risposta relativo al tasso λ=1.2 e p=0.1, che aumenta più del doppio.

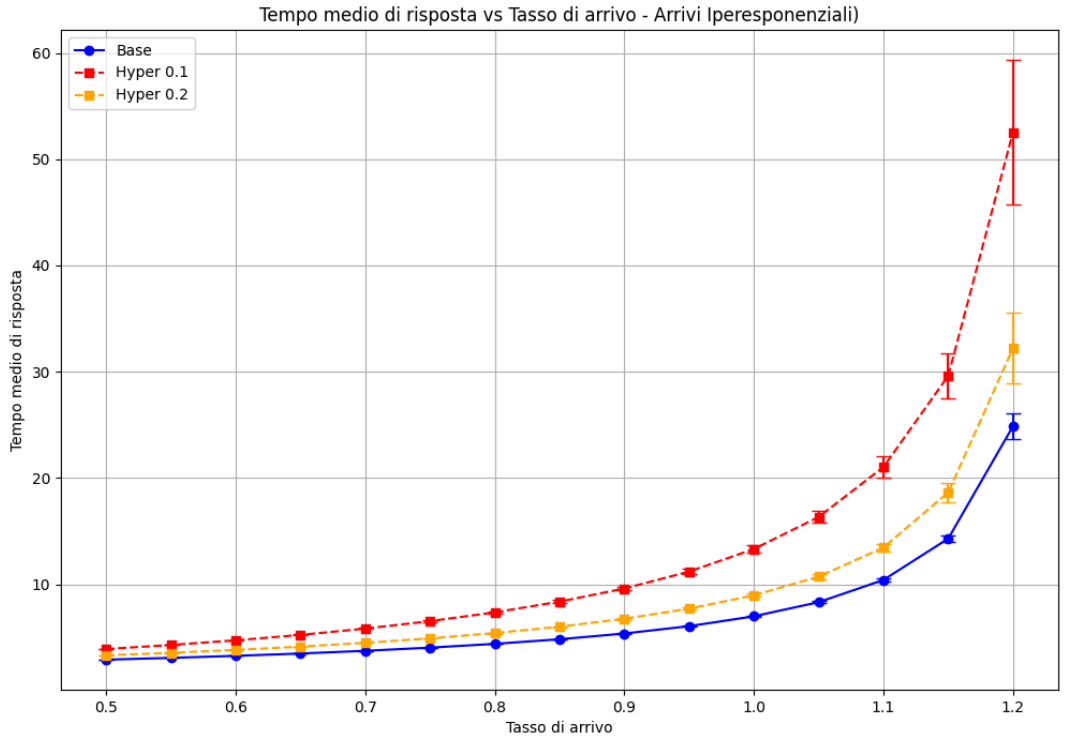


Grafico 4

12 Conclusioni

Gli obiettivi dello studio sono stati tutti raggiunti con risultati soddisfacenti.

Come espresso in precedenza, un'analisi più dettagliata richiederebbe delle fasce orarie con distribuzioni e tassi d’arrivo variabili per modellare almeno un intero giorno o una settimana. In questo modo sarebbe possibile esplorare diverse soluzioni in risposta a dei QoS, anch’essi da definire.

Per dei lavori futuri una possibilità di indagine potrebbe consistere in una diversa separazione delle parti elementari del sistema, ad esempio si potrebbe separare il database.
Invece, dal lato degli utenti si potrebbe indagare un ramo di rientro per quanto riguarda il Server B in modo da modellare l’aggiornamento del carrello e la navigazione nel catalogo.

Bibliografia

1. G. Serazzi, Performance Engineering, Springer Open Access