UNIVERSITÀ DEGLI STUDI DI ROMA
TOR VERGATA

FACOLTÀ DI INGEGNERIA
Corso di Laurea Magistrale in
Ingegneria informatica

Performance modelling of computer systems and networks

TITOLO PROGETTO:
“Analisi di un sistema di infrastruttura di installazioni di SIM card di una compagnia telefonica”

Studente:
Stefano Costanzo
Matricola numero:
0300955
Professoressa:
Vittoria De Nitto Personé

Sommario

1. Descrizione del sistema

Il presente progetto mira ad analizzare una struttura di infrastrutture, nello specifico un sistema di infrastrutture di installazione di SIM card telefoniche. Grazie alle conoscenze di un operatore nel settore è stato possibile ricavare le informazioni necessarie per avviare il progetto. Il sistema comprende tre processi (in gergo professionale “ambienti”) che sono **sviluppo**, **collaudo** e **produzione**, tutti concatenati ed indipendenti l’uno all’altro, speculari da un punto di vista funzionale ma contrapposti in termini di quantità di server (le fasi di sviluppo e collaudo sono provviste di meno risorse in termini di quantità di server rispetto alla fase di produzione che ne vanta quasi sette volte di più). Le fasi di sviluppo e collaudo, chiamate anche “ambienti di test”, permettono di creare vere e proprie simulazioni per assodare la presenza o assenza di eventuali problematiche che potrebbero emergere in seguito. Dunque, mentre per le prime due il procedimento è medesimo ma simulato, solo nella fase di produzione si agirà direttamente sul prodotto finale. Ciò porta il sistema di infrastrutture a far fronte a diverse difficoltà, per esempio: problematiche impossibili da replicare completamente negli ambienti di test, oppure complicazioni da un punto di vista prestazionale a causa dell’elevato gap dovuto alla differenza numerica della presenza di server. L’installazione dei software nasce da tre diverse esigenze: a causa di un’anomalia, un requisito o per un semplice aggiornamento. Una volta giunta la richiesta di installazione, questa attiva il sistema che prima di tutto analizza la problematica e sviluppa il software necessario grazie al gruppo di sviluppo. Dopo averlo sviluppato e testato nel loro ambiente, inviano il software al team di collaudo, che lo installa sulle proprie macchine e testa il risultato, che generalmente riguarda:

i. Testare se l’anomalia per cui è stato richiesto lo sviluppo sia stata risolta.
ii. Test di non regressione, permette di verificare se ciò che prima funzionava funziona ancora.
iii. Verificare che non siano state introdotte nuove anomalie.

Se entrambe le fasi vengono superate, il software potrà essere installato in produzione.
In caso contrario, se durante la fase di installazione o durante la fase di test vengono riscontrate problematiche, il software retrocede al gruppo di sviluppo, dove viene modificato per essere poi nuovamente trasferito al gruppo di collaudo.
L’installazione viene programmata in base alla sua priorità. Tale priorità dipenderà dall’impatto sul cliente (in questo caso il proprietario della SIM card), differenziando così 4 diversi livelli di priorità:

i. Livello 1 (“Very high”): riguarda installazioni urgenti e necessarie.
ii. Livello 2 (“High”): per installazioni importanti ed evidenti all’utente finale.
iii. Livello 3 (“Medium”): per installazioni trasparenti all’utente finale, nella maggior parte dei casi per implementazione di politiche di security o di requisiti minori.
iv. Livello 4 (“Low”): riguarda installazioni di minima priorità, spesso riguardanti documentazione.

Inoltre, tenendo sempre conto dell’impatto sul cliente, vengono scelte sia data che ora in base al possibile disservizio:

i. Se l’installazione non prevede disservizio solitamente si installa in orario di ufficio.

ii. Se l’installazione non prevede disservizio, ma solo un rallentamento, si cerca di installare ad orari con meno carico (verso l’ora di pranzo o la sera dopo le 22:00).

iii. Se l’installazione prevede un disservizio l’installazione avviene dopo le ore 2:00.

Come per la procedura di collaudo, anche nella produzione, dopo esser stato installato il software, vengono effettuati i test necessari e se non si riscontrano problematiche si può considerare risolta l’anomalia, differentemente si torna alle fasi precedenti.
Ognuno dei tre ambienti si avvale di un personale team working che focalizza la propria attenzione esclusivamente sull’ambiente di cui fa parte. Nel complesso, l’ambiente di sviluppo conta circa 8-9 persone che formano team working da 3-4 persone ciascuno, le quali sono specializzate su determinati elementi. Infatti, una volta riscontrata un’anomalia, questa viene suddivisa in base a tali specializzazioni. L’ambiente di collaudo, invece, si avvale di circa 7-8 persone, divise in un gruppo di installazione ed un gruppo di test. Infine, l’ambiente di produzione comprende circa 16-18 persone, ed anche in questo caso il gruppo viene scisso in team di installazione e di test. In questo caso il gruppo di installazione è composto da sistemisti che gestiscono tutte le macchine garantendone il funzionamento, mentre il team di test sovraintende anche al troubleshooting che, nell’ipotesi in cui siano presenti problematiche funzionali, permette di analizzare e verificare dove e come si verifica tale problema prima ancora di dichiarare l’anomalia.

2. Abstraction

- Job: Installazione
- Servente: Team di lavoro
- Algoritmo: Next event simulation
- Tipologie di job: suddivisione in base al livello di priorità (1 a 4)
- Selezione job nel centro: Prioritaria senza preemption. FIFO all’interno della stessa coda di attesa.

3. Modello concettuale

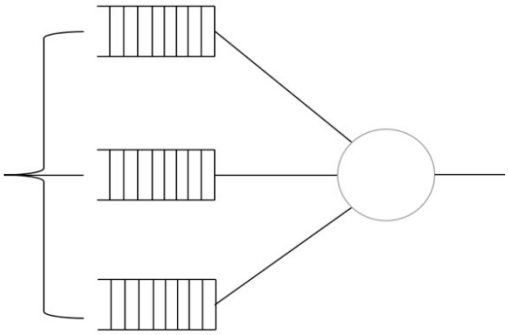
Per la modellazione del sistema è stata individuata la divisione nei 3 centri principali: Sviluppo, Collaudo, Produzione. Di seguito un'iniziale astrazione del flusso del job.

IMG1: Astrazione iniziale

1) Sviluppo

Viene modellato come un centro a 3 code di priorità e un servente.
Le code di priorità dividono l'attesa fra le priorità 2, 3 e 4, mentre la priorità massima seguirà un altro flusso diretto verso la Produzione. Il servente rappresenta l'intero gruppo di sviluppo, dal momento che i due team di lavoro possono lavorare contemporaneamente allo stesso job.

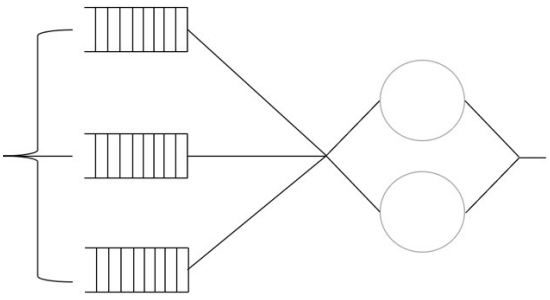
IMG2: Sviluppo



2) Collaudo

Come il precedente, anche questo è stato modellato come un centro a 3 code ma con due serventi anziché uno, essendo questi impossibilitati a lavorare sullo stesso job contemporaneamente.

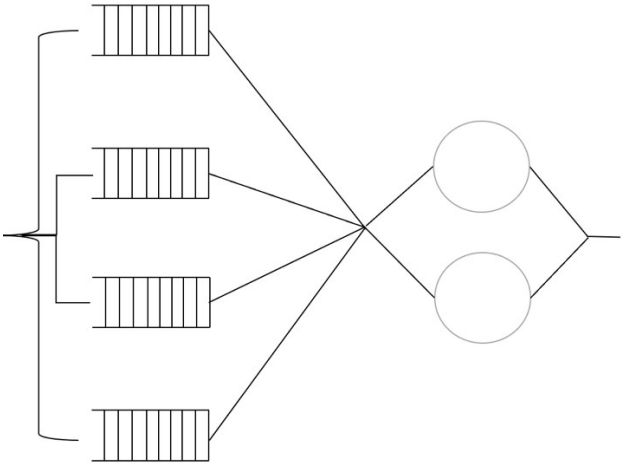
IMG3: Collaudo



3) Produzione

Viene modellato come un centro a 4 code e 2 serventi. In ingresso subentrano anche i job di priorità massima. Per i due serventi si applica lo stesso discorso del Collaudo essendo questa astrazione più appropriata nel dominio di analisi.

IMG4: Produzione



A questi, sono stati aggiunti e sviluppati altri 3 centri a servente a coda singola:

4) Fast Development-Test

Rappresenta il team di lavoro “straordinario” per lo sviluppo e il testing rapido dei job a priorità 1.

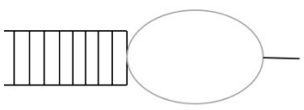
5) Delay Production

Si colloca fra il Collaudo e la Produzione e permette di astrarre l’attesa e il setup delle macchine in produzione per i job di priorità 2 (“high”). Necessariamente i job non trasparenti agli utenti finali, che quindi possono provocare un disservizio nell’utilizzo del sistema, devono essere rimandati in orari di minor traffico (ora di pranzo o ore notturne) o hanno bisogno di un setup particolare delle macchine.

6) Rollback

Si colloca fra Produzione e Sviluppo. Astrae I feedback della produzione allo sviluppo che a causa dell’installazione necessitano di un rollback delle macchine di produzione.

IMG5: Serventi a coda singola

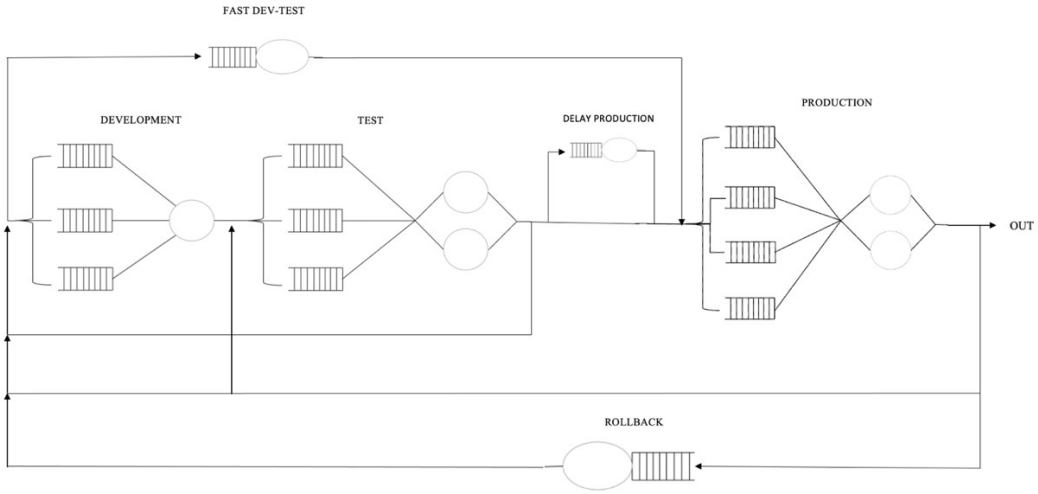


Feedback:
- Dal Collaudo allo sviluppo.
- Dalla Produzione al Test.

- Dalla Produzione allo Sviluppo.
- Dalla Produzione allo sviluppo tramite Rollback.

Di seguito il sistema completo:

IMG6: Sistema intero



Stato:

Si definisce lo stato del sistema come l'insieme degli stati di ogni servente. In particolare, nello stato di ogni servente verranno memorizzati i job in attesa, i job in servizio, lo stato del servente (libero o occupato), il numero di arrivi e il numero di partenze.
Per quanto riguarda gli eventi si distinguono due tipologie: arrivo di un'installazione, che corrisponde all'ingresso di un job nel sistema, e il completamento da parte di un servente. Il completamento di un job da parte di un servente corrisponde all'ingresso dello stesso job in un altro centro della rete, che verrà definito sulla base delle probabilità di routing. La gestione degli eventi segue le linee guida della simulazione next-event: tutti gli eventi che devono essere processati sono ordinati in una lista di eventi ordinata temporalmente. Chiaramente, l'elaborazione dell'evento potrebbe generarne altri che verranno inclusi nella struttura dati che mantiene tutti gli eventi.

4. Modello specifiche

Distribuzioni
Esponenziale

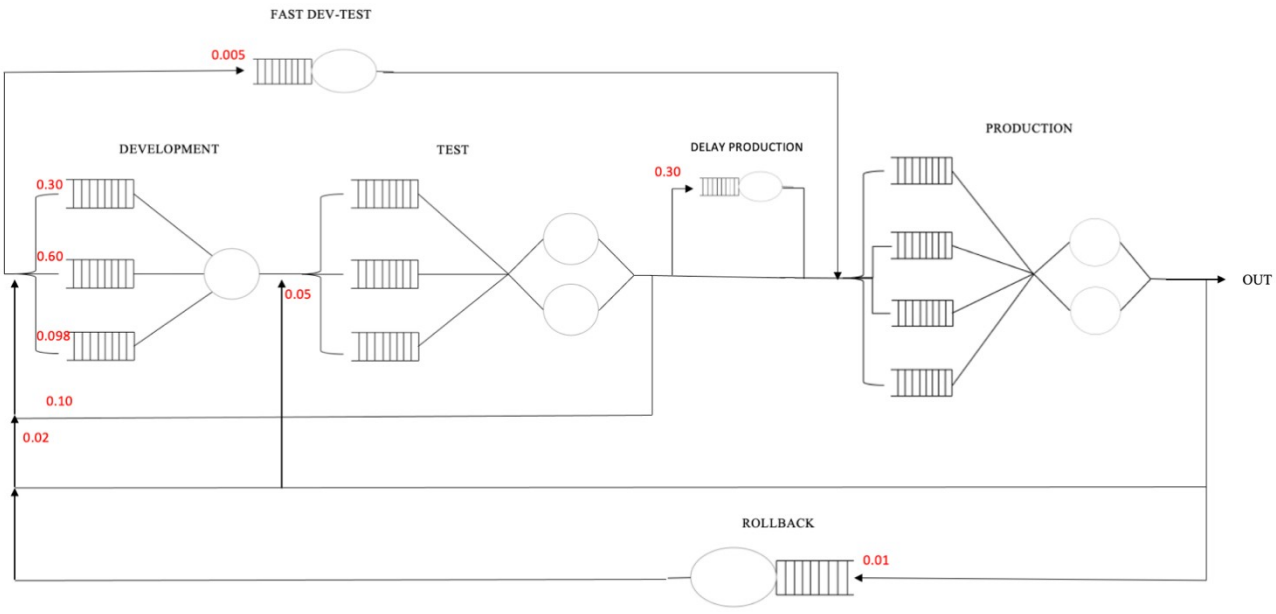
Probabilita routing
Probabilità feedback Collaudo verso Sviluppo: 10%
Probabilità feedback Produzione verso Collaudo: 5%
Probabilità feedback Produzione verso Sviluppo: 2%
Probabilità feedback Produzione verso Sviluppo con rollback: 1%

Tassi di arrivo
Il numero medio di arrivi nel sistema su base giornaliera è 4 job/giorno
Questa viene suddivisa per le varie priorità come segue:
- probabilità job priorità 1: 0.5%
- probabilità job priorità 2: 30%
- probabilità job priorità 3: 60%
- probabilità job priorità 4: 9.5%

Tempi di servizio
E[Si] sviluppo : 5 job/giorno
E[Si] collaudo : 4 job/giorno
E[Si] produzione : 8 job/giorno
E[S] delay Prod. : 6 job/giorno
E[S] rollback : 2 job/giorno
E[S] fast devel-test : 12 job/giorno

QOS
1) Job priorità 1: E[Ts] < 5 ore
2) Job priorità 2: E[Ts] < 24 ore
3) Job priorità 3: E[Ts] < 3 giorni

IMG7: Sistema con probabilità di routing



5. Modello computazionale

Generatori pseudocasuali

Per la generazione dei tempi di servizio è stato utilizzato un generatore pseudocasuale che è in realtà la conversione in java del file C Rngs.c . Si tratta di un generatore multistream i cui parametri utilizzati per la generazione dei valori sono : MODULUS = 2147483647, MULTIPLIER = 48271, A256 = 22925, seme DEFAULT = 123456789 ,STREAMS = 256. Sono previste diverse funzionalità sia per piantare un seme, cambiare stream sia per generare nuovi valori secondo una distribuzione normale , pareto esponenziale ecc.
Per ogni evento si è utilizzato un seed diverso per la generazione dei numeri pseudocasuali. Sono stati adoperati 14 seed:
7 per gli eventi del sistema: 1 per arrivi, 6 per i servizi di ogni blocco.
7 per le corrispettive probabilità di routing (*seed_routing = seed_eventi + 7*): Ogni evento nel sistema necessità del calcolo di una probabilità di routing per l'ingresso del job in un altro blocco o per la sua terminazione.

Strutture principali

- Stato: Completa caratterizzazione del sistema. 3 multiserver a coda multipla e 3 server a coda singola.
- Lista di eventi: Lista contenente i prossimi eventi in ordine temporale
- Configurazione: Struttura che descrive la topologia di rete da usare. Parametri ingresso e di uscita, probabilità di routing e numero servers nei centri.
- Job: Struttura per la modellazione dell'installazione.
- Statistica: Output del sistema

IMG8: Next event algorithm

```
    // START SIMULATION
    if(clock == 0){
        // Plant seed
        PlantSeeds(conf->seed);
        scheduleEvent(arrival, -1, -1);
    }

    while(clock < conf->close_the_door && stats->jobs_output < conf->totalJobs){
        // 2) GET EVENT
                e = calqueue_get(eventList);

        // 3) ADVANCE
        clock = e->time;

        // 4) PROCESS AND SCHEDULE
        if(e->type == arrival) exec_arrival(e);
        else exec_serv(e);

        // 5) REMOVE EVENT
            free(e);
    }

    // 6) GENERATE STATISTICS
    finalUpdStats();

    *clk = clock;
}
```

6. Verifica

L’obiettivo di questa fase è verificare che il software rispetti le specifiche di progetto e produca risultati in accordo con la teoria. L’approccio scelto si basa su una tecnica di test definita *partizionamento del dominio*, dove si identificano tutti i sottosistemi possibili da simulare per dimostrare che in ogni configurazione il modello implementativo produrrà risultati concordi con la teoria. Di conseguenza, partendo da una base di software verificata tramite questo approccio, è possibile considerare e simulare una rete complicata liberamente con una ragionevole confidenza che questa restituisca risultati corretti. I valori che la simulazione vuole verificare in questa fase sono: tempi di attesa, tempi di servizio, tempo di risposta, popolazione media in coda, popolazione media nei serventi e popolazione media nel centro.

Per poter eseguire un confronto valido è necessario definire dei valori medi e i rispettivi intervalli di confidenza: A tal proposito si è optato per un’analisi batch means (b = 512, k = 1024) ed intervalli di confidenza del 95%.
Di seguito i risultati per i vari centri.

IMG9: Confronto modello analitico con simulazione

	E[Tq]	E[s]	E[Ts]	w	x	l	lamda'
C1 Simul.	54.2713h +/- 2.1587h	4.7907h +/- 0.0124h	59.0620h +/- 2.1610h	10.7514 +/- 0.4538	0.9210 +/- 0.0028	11.6734 +/- 0.4555	4.6247 +/- 0.0130
C1 Analit.	55.852h	4.8h	60.652h	10.715	0.9208	11.6358	4.604298
C2 Simul.	3.3247h +/- 0.0656h	5.9837h +/- 0.0146h	9.3084h +/- 0.0738h	0.6792 +/- 0.0145	1.2074 +/- 0.0045	1.8866 +/- 0.0179	4.8431 +/- 0.0139
C2 Analit.	3.422h	6h	9.422h	0.6874	1.2053	1.8927	4.821254
C3 Simul.	0.2363h +/- 0.0052h	2.9904h +/- 0.0077h	3.2267h +/- 0.0108h	0.0433 +/- 0.0010	0.5440 +/- 0.0021	0.5873 +/- 0.0027	4.3656 +/- 0.0123
C3 Analit.	0.2392h	3h	3.2392h	0.0433	0.5434	0.5867	4.347129
C4 Simul.	1.0932h +/- 0.0242h	3.9840h +/- 0.0190h	5.0772h +/- 0.0367h	0.0604 +/- 0.0015	0.2164 +/- 0.0015	0.2768 +/- 0.0026	1.3042 +/- 0.0063
C4 Analit.	1.0454h	4h	5.0454h	0.0542	0.2071	0.2613	1.24316
C5 Simul.	0.1385h +/- 0.0367h	11.0507h +/- 0.3548h	11.1892h +/- 0.3603h	0.0004 +/- 0.0001	0.0214 +/- 0.0008	0.0218 +/- 0.0008	0.0435 +/- 0.0011
C5 Analit.	0.2666h	12h	12.2666h	0.00048	0.0217	0.0222	0.0434
C6 Simul.	0.0000h +/- 0.0000h	0.7976h +/- 0.0643h	0.7976h +/- 0.0643h	0.0000 +/- 0.0000	0.0007 +/- 0.0001	0.0007 +/- 0.0001	0.0080 +/- 0.0005
C6 Analit.	/	2h	2h	/	0.00067	0.00067	0.008

Come possiamo vedere i risultati ottenuti sono congruenti con il modello analitico. Solo i centri 5 e 6 non riportano risultati accurati, probabilmente a causa della bassa probabilità di ingresso negli stessi e perciò della loro maggior volatilità nel calcolo statistico.

Consistenza

Per la verifica di consistenza del modello sono state applicate per ogni blocco le seguenti formule:

A) E[Ts] = E[Tq] + E[s]
B) l = q + x
C) Lamda’ = Lambda * Prob_feedback

Come visibile dalla figura le proprieta sono sempre verificate.
Lambda’ è stato calcolato tenendo conto delle probabilità di routing in ingresso nei vari blocchi. Apportando una leggera approssimazione i risultati sono coerenti con gli attivi del modello simulato.

7. Validazione

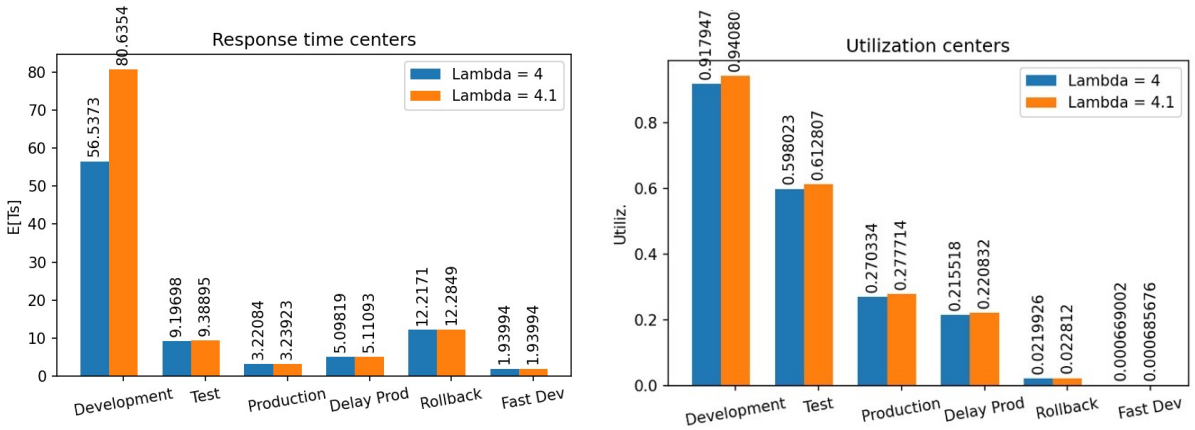
Se da un lato è stato possibile individuare i dati per la realizzazione delle specifiche del progetto, dall’altro è stato complicato trovarne di utili per la validazione. Questa fase tenderà quindi a verificare che, cambiando minimamente alcuni parametri del sistema, il modello possa dimostrare un comportamento verosimile che rispecchi la realtà.

I test che verranno eseguiti sono sostanzialmente 4:

- 1) Aumento tassi di ingresso

Aumentando il carico di lavoro nel sistema, il tempo di risposta totale e l’utilizzazione aumentano: Lo Sviluppo ne risente maggiormente avendo un utilizzazione prossima alla saturazione.
Lambda: 4 → 5

IMG10: Aumento tassi ingresso

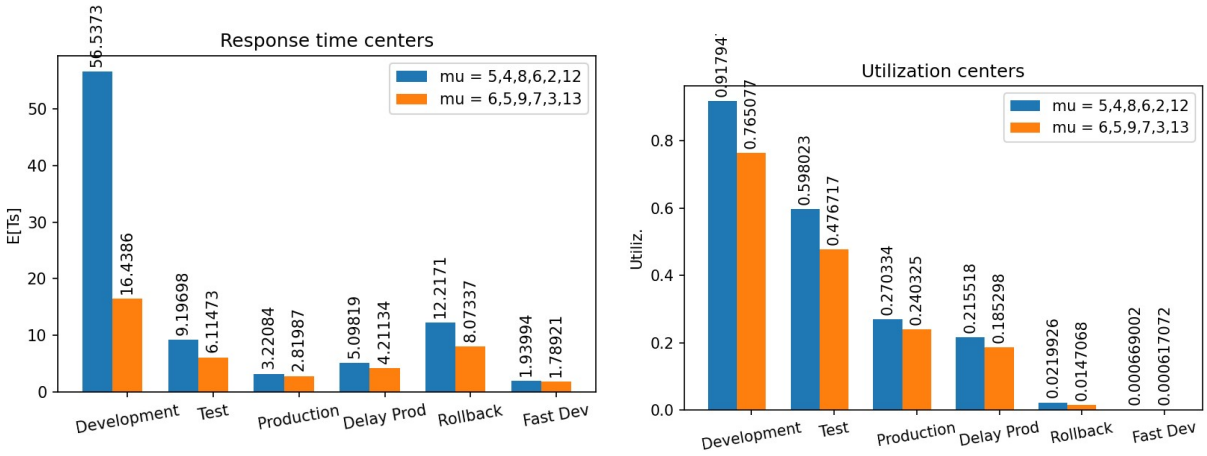


- 2) Diminuzione tempi di servizio

Discorso inverso per i tempi di servizio: Il carico del sistema viene smaltito più velocemente.
Tassi servizio:

dev: 5 → 6, test: 4 → 5, prod: 8 → 9
delayProd: 6 → 7, rollback: 2 → 3, fastdev: 12 → 13

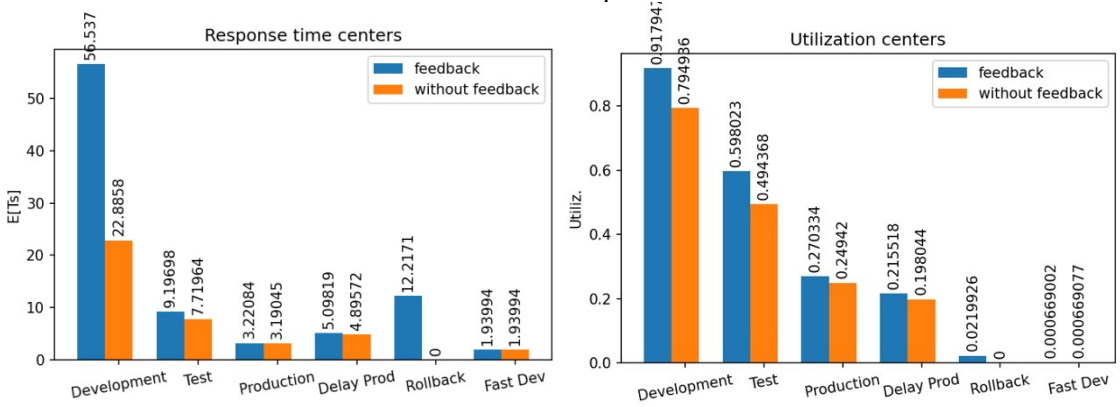
IMG11: Aumento tassi di servizio



- 3) Zero feedback

Come volevasi dimostrare togliendo I feedback il sistema guadagna notevolmente in prestazioni.
Caso di analisi: Azzeramento delle probabilità di routing per I feedback

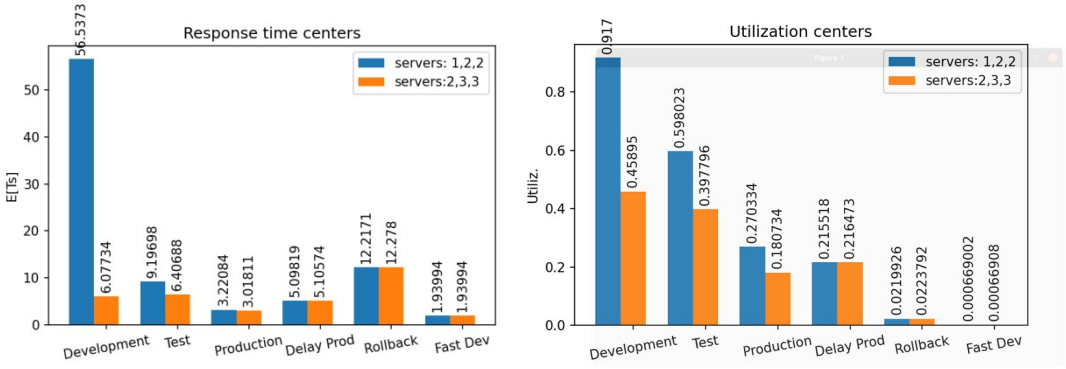
IMG12: Zero feedback



- 4) Aumento numero server in Sviluppo, Collaudo, Produzione

E’ facilmente intuibile che all’aumento del numero di server il sistema subisca un forte speed-up delle prestazioni, molto maggiore ai casi precedenti.
Caso analisi:
Numero serventi in sviluppo: 1 - > 2
Numero serventi in collaudo: 2 - > 3
Numero serventi in produzione: 2 - > 3

IMG13: Aumento numero serventi

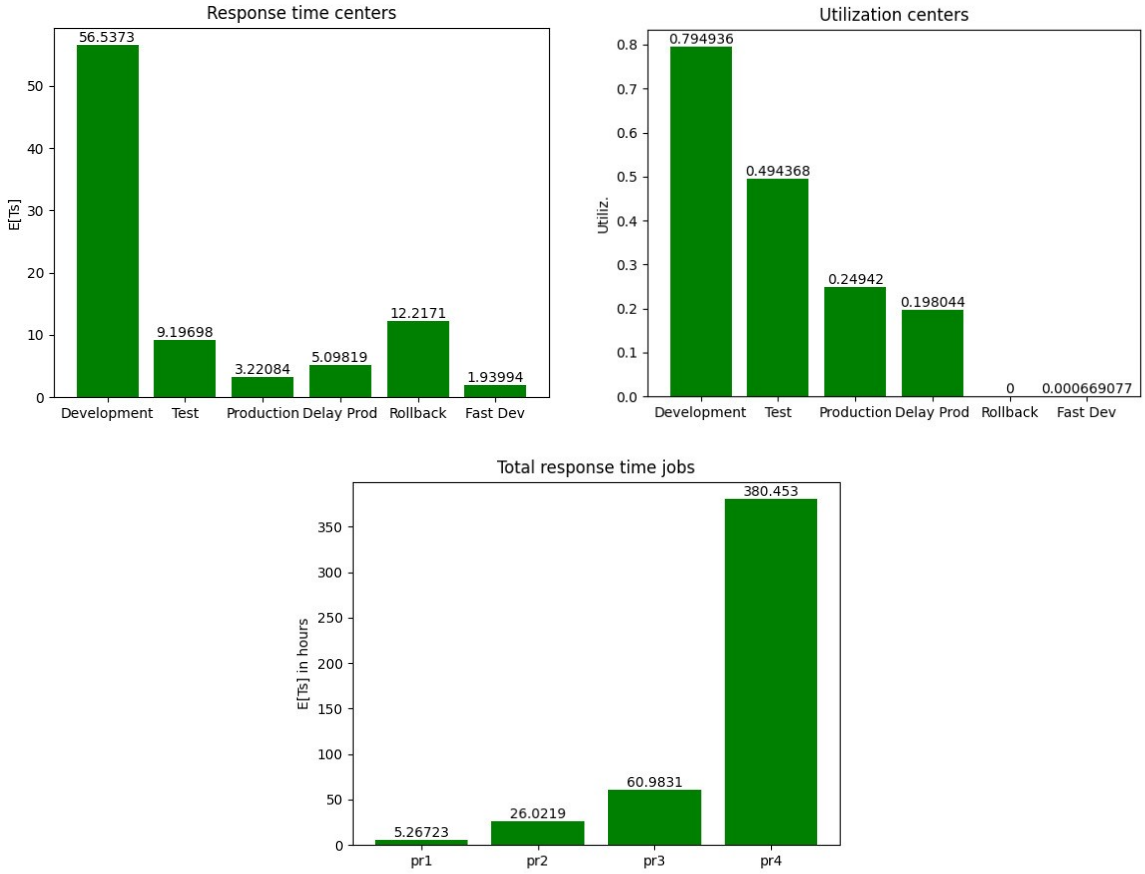


7. Analisi orizzonte infinito

L’obiettivo del progetto è quello di far convergere i tempi di riposta medi dei vari centri ai tempi analitici. Appare evidente che, poichè l’obiettivo richiede il raggiungimento di una convergenza, ha molto più senso realizzare una simulazione a orizzonte temporale infinito. Nello specifico è stata condotta un un’unica simulazione a lungo orizzonte temporale con condizione di arresto a completamento di 300 000 jobs.

Di seguito vengono riportati i valori del tempo di risposta, utilizzazione dei 6 centri di servizio e tempi medi globali dei jobs per priorità. Quest’ultimo grafico ha un notevole risonanza nel nostro caso di studio, i QOS impongono limiti a questi jobs

IMG14: Analisi steady state



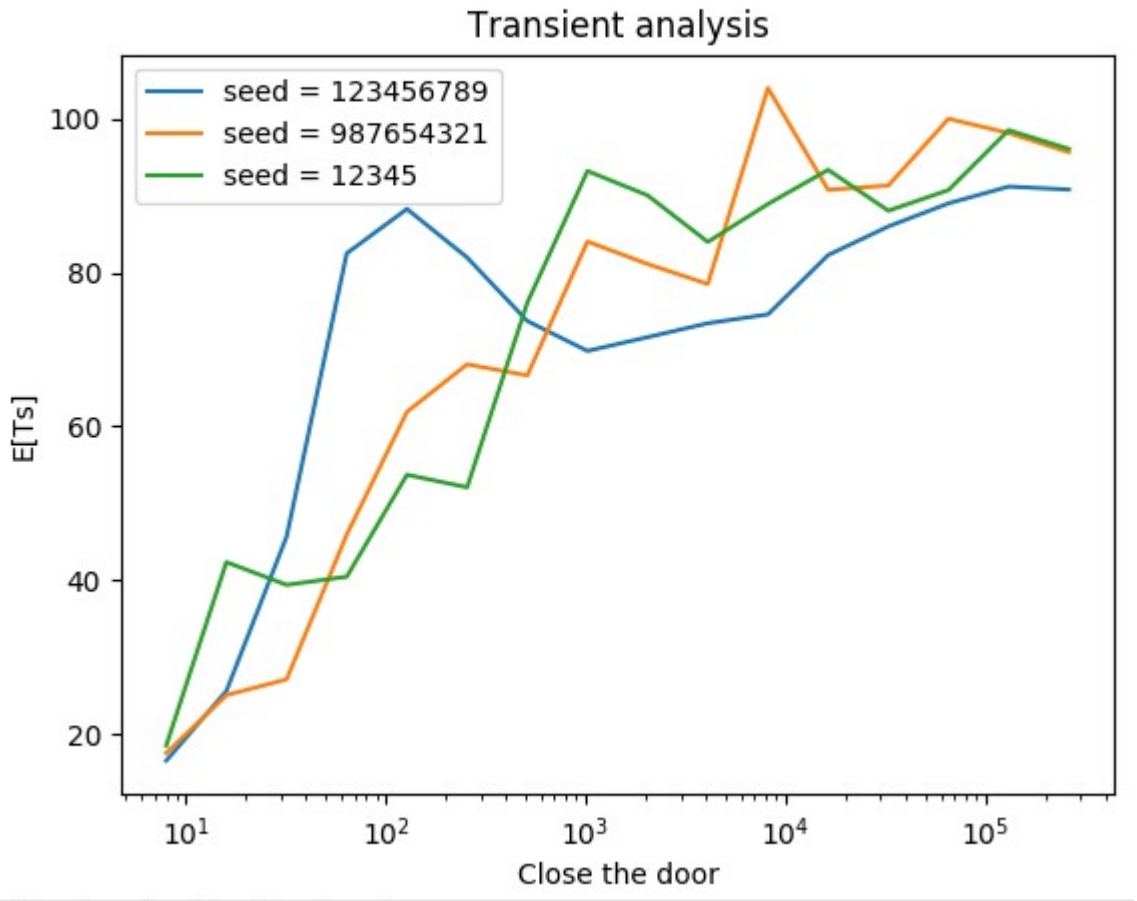
Il sistema non rispetta il QOS1 e QOS2: Il tempo di risposta dei job di priorità 1 e 2 sono rispettivamente superiori a 4 ore e 24 ore. E’ necessario un miglioramento del sistema.

Job priorità 1 E[Ts] = 5.26h > 5h [QOS1]
Job priorità 1 E[Ts] = 26h > 24h [QOS2]
Job priorità 1 E[Ts] = 60.9h < 72h [QOS3] V

8. Analisi orizzonte finito

L’analisi transiente sebbene meno rilevante per il soddisfacimento dei QOS ha comunque importanza nello studio di particolari e temporanee condizioni in cui può trovarsi il sistema. Questa viene eseguita mantenendo le condizioni del sistema iniziale identiche per ogni ripetizione e variando il seme della funzione pseudocasuale.
I seed considerati sono 123456789, 987654321, 12345.
Di seguito un grafico del tempo di risposta del sistema all’aumentare del tempo di simulazione, ovvero del parametro close_the_door.

IMG15: Finite horizon analysis



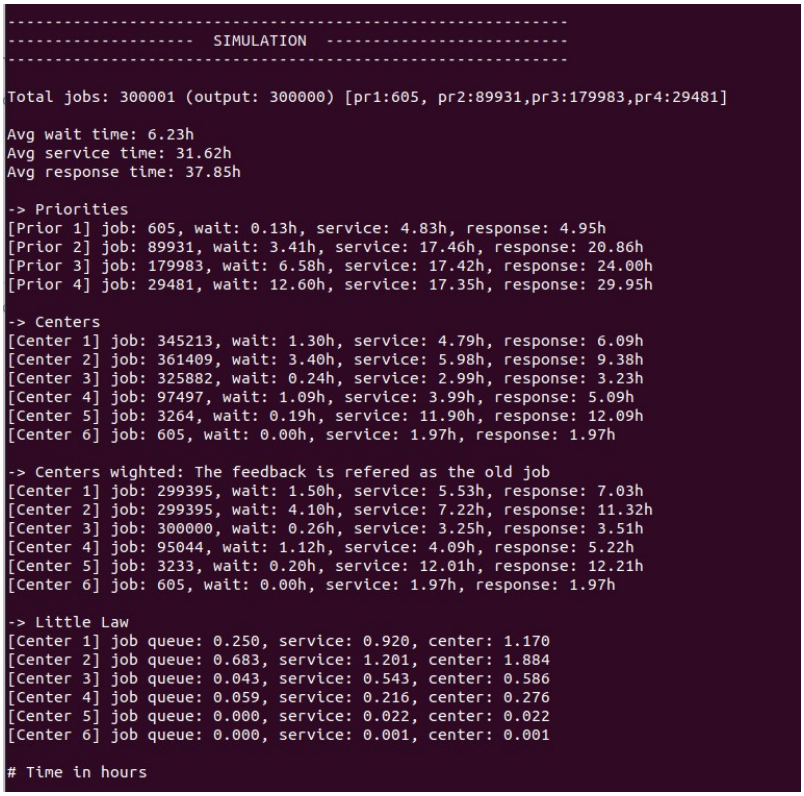
Il sistema presenta unaa forte variabilità fino a 10^3 giorni di esecuzione (circa 2,7 anni). Tende a convergere solo per tempi lunghissimi ad un E[Ts] = 92hr

9. Conclusioni

con proposta di modello migliorativo

L’analisi condotta ha permesso di individuare alcune criticità dell’infrastruttura.
Il sistema necessità di un miglioramento per il raggiungimento dei QOS. I centri di fast development e rollback hanno un utilizzazione minima, rispettivamente di 7 * 10^-4 e 2 * 10^-2.
Mentre il centro di Sviluppo è il collo di bottiglia del sistema: Ha tempi di attesa molto maggiori rispetto agli altri centri con un tempo di attesa maggiore di 55 ore e un’utilizzazione di U = 0.92.

Come modello migliorativo si propone l’aggiunta di un secondo server in Sviluppo.



I tempi di attesa, in primo luogo del centro di Sviluppo, sono abbattuti fortemente portando i tempi di risposta da un precedente E[Ts] =92,62hr a 37.85hr, dove solo 6.23hr per il tempo di attesa. Ancora più importante è il raggiungimento dei tempi di risposta dei QOS.

[QOS1] job priorità 1 E[Ts] = 4.95h < 5h.
[QOS2] job priorità 2 E[Ts] = 20.86h < 24h.
[QOS3] job priorità 3 E[Ts] = 24h < 72h