Heavy traffic limit for a tandem queue with identical service times

Simulazione di Sistemi

Giuseppe Di Maria

ALMA MATER STUDIORUM A.D. 1088
UNIVERSITÀ DI BOLOGNA

Contents

1 Introduzione

Il presente progetto si focalizza sull'analisi delle prestazioni di un sistema di coda basato su simulazioni utilizzando OMNeT++. L'obiettivo principale è valutare il tempo di risposta del sistema e il fattore di utilizzo dei singoli server. Il sistema di coda in esame è composto da diversi moduli, tra cui generatori di lavoro, code di attesa e server. L'analisi delle prestazioni viene effettuata attraverso la raccolta di dati di simulazione e l'elaborazione statistica delle misure ottenute.

2 Tool Utilizzati

Per la realizzazione del progetto, sono stati impiegati diversi strumenti e tecnologie. In particolare, è stato utilizzato OMNeT++, un framework di simulazione ad eventi discreti ampiamente utilizzato per lo sviluppo di modelli di rete e sistemi complessi. OMNeT++ fornisce una vasta gamma di funzionalità per la modellazione e la simulazione di sistemi di coda, consentendo di esaminare il comportamento dinamico e le prestazioni del sistema in uno scenario controllato.
Inoltre, per l'elaborazione dei dati di simulazione e la generazione dei grafici, è stato impiegato il linguaggio di programmazione Python. Python offre un'ampia gamma di librerie e strumenti per l'analisi dati, tra cui Pandas per la manipolazione e l'analisi dei dati tabulari e Matplotlib per la creazione di grafici e visualizzazioni. L'utilizzo di Python ha consentito di automatizzare l'analisi dei risultati di simulazione e la generazione di grafici descrittivi, facilitando l'interpretazione e la presentazione dei dati ottenuti durante le simulazioni.
L'impiego congiunto di OMNeT++ e Python ha permesso di realizzare un processo di simulazione efficace e una successiva analisi delle prestazioni dettagliata. Questo ha consentito di valutare le prestazioni del sistema di coda in esame e fornire informazioni utili per l'ottimizzazione e il miglioramento delle sue caratteristiche. Nelle sezioni successive, saranno presentati nel dettaglio il modello di simulazione sviluppato, le metriche di prestazione analizzate e i risultati ottenuti.

3 Modello

3.1 Descrizione

Il modello di simulazione realizzato rappresenta un sistema composto da un generatore di lavoro, code di attesa e server relativi. Il sistema è stato implementato utilizzando il framework OMNeT++, che fornisce gli strumenti necessari per la creazione di modelli di simulazione ad eventi discreti.

Il sistema di coda è costituito da una sorgente di lavoro, responsabile della generazione di nuovi job da instradare nel sistema. I job generati vengono quindi inviati alle code di attesa, dove vengono gestiti secondo una determinata politica di instradamento. Nel modello, sono state implementate diverse politiche di

instradamento, come ad esempio la politica FIFO (First-In-First-Out), consentendo di valutare le prestazioni del sistema in diverse configurazioni.

I job presenti nelle code di attesa vengono poi inviati ai server per essere elaborati. I server rappresentano le risorse di calcolo del sistema e sono responsabili dell'esecuzione dei job. Nel modello, è possibile specificare il tempo di servizio di ciascun server, che determina la durata necessaria per elaborare un job. Durante la simulazione, i job vengono prelevati dalle code di attesa in base alla politica di instradamento configurata e vengono assegnati ai server disponibili.

Durante l'esecuzione del sistema, vengono raccolti diversi dati di interesse per l'analisi delle prestazioni. Ad esempio, viene misurato il tempo di risposta del sistema, che rappresenta il tempo trascorso da un job dall'arrivo al completamento. Questo dato fornisce informazioni importanti sulle prestazioni complessive del sistema. Inoltre, viene calcolato il fattore di utilizzo dei singoli server, che rappresenta la percentuale di tempo in cui un server è occupato rispetto al tempo totale di simulazione.

Attraverso la simulazione del modello e l'analisi dei dati raccolti, è possibile valutare le prestazioni del sistema di coda in diverse condizioni di carico e con diverse politiche di instradamento. Questo consente di identificare eventuali criticità o inefficienze nel sistema e di adottare misure correttive per migliorarne le prestazioni complessive.

Nelle sezioni successive della documentazione, saranno presentati i dettagli implementativi del modello di simulazione, le metriche di prestazione analizzate e i risultati ottenuti durante le simulazioni.

I moduli coinvolti sono:

- Source: Modulo responsabile della generazione dei job da inviare nel sistema di coda. Genera in modo periodico i job e li inoltra alla coda successiva.
- PassiveQueue: Modulo che rappresenta una coda di attesa passiva per i job. I job ricevuti vengono memorizzati in questa coda fino a quando non vengono assegnati a un server disponibile per l'elaborazione.
- Server: Modulo che rappresenta un server nel sistema di coda. Riceve i job dalle code di attesa e li elabora secondo un determinato tempo di servizio. Al termine dell'elaborazione, invia il job al modulo successivo o lo scarta, a seconda delle politiche implementate.
- Router: Modulo responsabile dell'instradamento dei job dalle code di attesa ai server disponibili. Implementa diverse politiche di instradamento, come FIFO o LIFO, per determinare l'ordine di assegnazione dei job ai server.
- Sink: Modulo finale del sistema di coda, che raccoglie i job completati. Misura il tempo di risposta dei job e raccoglie statistiche sul tempo di vita dei job arrivati. Inoltre, può emettere segnali relativi al tempo di risposta del sistema.

3.2 Parametri

Le simulazioni sul modello sono state eseguite su ogni possibile combinazione dei seguenti parametri:

- λ: Rappresenta il tasso di arrivo dei job nel sistema di coda. I suoi valori determinano la frequenza con cui i job vengono generati e inviati al sistema di coda.
- $S1$: Rappresenta il tempo di servizio dei job nei server. È stato utilizzato un modello di distribuzione esponenziale negativa con una media di 1/1. I valori di 1 utilizzati nella simulazione sono stati 3.0 e 4.0. Questi valori influenzano la velocità con cui i job vengono elaborati dai server.
- Si: Rappresenta il tempo di servizio dei job nelle code di attesa. È stata utilizzata una distribuzione uniforme con intervallo [a, b]. I valori di [a, b] utilizzati nella simulazione sono [1, 6] e [2, 4]. Questi valori definiscono l'intervallo di tempo in cui i job rimangono nelle code di attesa prima di essere assegnati ai server.
- p: Rappresenta la probabilità di selezionare una determinata politica di instradamento dei job. Nella simulazione, sono stati utilizzati i seguenti valori per p: 0.2, 0.4, 0.8. Questi valori determinano la preferenza per una politica di instradamento rispetto alle altre e influenzano l'ordine di assegnazione dei job ai server.

I valori configurabili per i parametri sono:

- λ: 2.0, 1.4, 1.2, 1.0
- $S1$: Esponenziale negativa di media $1/\mu 1$ (con $\lambda = 2.0...1.0$, $\mu 1 = 3.0$, $\mu 1 = 4.0$)
- Si: Distribuzione uniforme in $[a, b]$ ($[a, b] = [1, 6]$, $[2, 4]$)
- p: 0.2, 0.4, 0.8

La scelta di diversi valori per questi parametri consente di esplorare e analizzare il comportamento del sistema di coda in diverse configurazioni e condizioni operative.
Sono stati generati quindi file di configurazione della simulazione, tante quante sono le possibili combinazione dei parametri. In totale, sono stati utilizzati 72 diversi file di configurazione.

4 Implementazione

Per l'implementazione sono stati utilizzati i moduli implementati dalla libreria queueinglib, apportando le opportune modifiche ai nodi PassiveQueue, Server e Sink.

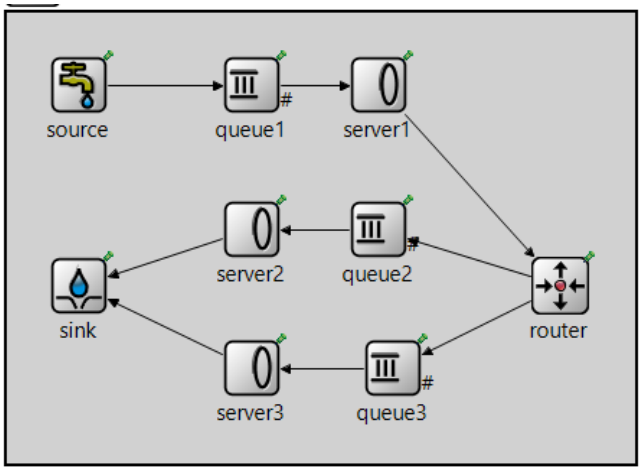

Figure 1: Heavy Traffic Limits Network

4.1 Source

Il modulo 'Source' nel progetto svolge il ruolo di generatore di job. È responsabile della creazione e invio dei job nel sistema di simulazione. Il modulo 'Source' viene configurato tramite i parametri nel file di configurazione .ned. Questi parametri includono il numero di job da generare, il tempo di interarrivo dei job e il periodo di tempo in cui il modulo genera job.
La classe 'Source contiene gli attributi 'startTime', 'stopTime', 'numJobs' e 'lambda', che sono configurabili. Il metodo 'initialize()' inizializza il modulo e programma il primo evento di generazione del job al tempo di 'startTime'.

4.2 PassiveQueue

Il modulo PassiveQueue implementa una coda passiva, che significa che si limita a memorizzare i messaggi in ingresso senza elaborarli attivamente.
I messaggi vengono inseriti nella coda e possono essere prelevati da un modulo successivo quando quest'ultimo è pronto per elaborarli. La coda passiva è asincrona, il che significa che non impone ritardi o attese al modulo che invia i messaggi.
La coda passiva memorizza i job in arrivo senza elaborarli attivamente, consentendo al modulo successivo (il primo servente) di prelevare i job dalla coda quando è pronto per elaborarli.
La sua funzione principale è gestire l'instradamento dei job provenienti dai moduli precedenti verso i server appropriati. Ogni server nel sistema ha una coda passiva ad esso associata, creata attraverso l'utilizzo del modulo 'PassiveQueue'.
Il modulo 'PassiveQueue' viene configurato con diversi parametri, come la capacità della coda (illimitata), l'algoritmo di invio dei job alla coda (come priorità, casuale, round-robin o minimo ritardo) e il tipo di ordinamento FIFO.

Inoltre, il modulo tiene traccia di varie statistiche, come la lunghezza della coda, il tempo di attesa nella coda e il numero di job persi.

4.3 Server

Il modulo server è responsabile della gestione delle code di input (PassiveQueue) e dell'elaborazione dei job. Vengono registrati i segnali e vengono creati la strategia di selezione e il messaggio di fine servizio

Durante la gestione dei messaggi, il server controlla se il messaggio ricevuto è il messaggio di fine servizio o un nuovo job:

- Se è il messaggio di fine servizio, il server invia il job al suo gate di output e richiede un nuovo job dalle code di input;
- Se è un nuovo job, il server imposta il job in servizio, schedula il messaggio di fine servizio e verifica se la scadenza del job è stata superata.

In caso contrario (se la scadenza del job non è stata superata), il server imposta lo stato di occupazione e continua l'elaborazione del job.

4.4 Sink

Il modulo 'sink' è responsabile della ricezione dei job, della raccolta delle statistiche e della distruzione dei pacchetti ricevuti. Durante la gestione dei messaggi, il modulo 'sink' calcola il tempo di risposta del job, emette il segnale corrispondente e distrugge il messaggio del job. Se l'opzione per mantenere i job è attiva (keepJobs = true), i messaggi non vengono eliminati.

Questo modulo 'sink' viene utilizzato per terminare i job e raccogliere statistiche sui tempi di vita e i tempi di risposta dei job nel sistema simulato.

5 Statistiche

L'obiettivo era quello di simulare il modello, raccoglierne e analizzarne le statistiche al fine di convalidarne il funzionamento, in modo da poterlo allineare agli altri modelli analitici di reti a coda. Per ogni possibile combinazione di parametri delle configurazioni di simulazione, sono stati calcolati le stime puntuali e relativi intervalli di confidenza:

- mediana della distribuzione del tempo di risposta del sistema
- mediana della distribuzione del tempo di risposta di ogni singolo servente
- tempo medio di permanenza nel sistema dei job
- fattore di utilizzo dei singoli server

Per ogni possibile configurazione sono state eseguite 20 run, per un tempo complessivo di 100.000 secondi.

5.1 Transiente Iniziale

Nel momento in cui un sistema inizia da poco ad essere operativo, tale sistema è fortemente influenzato dallo stato iniziale e dal tempo trascorso prima dell'attivazione. Questa condizione del sistema 'e detta transitoria. Trascorso un lasso di tempo iniziale il sistema si stabilizza e raggiunge condizioni stazionarie. Durante gli esperimenti, visualizzando i risultati ottenuti, è stato notato un comportamento anomalo nella fase iniziale. Nella figura seguente, la parte evidenziata in grigio dai rettangoli rappresenta il periodo del transiente iniziale.

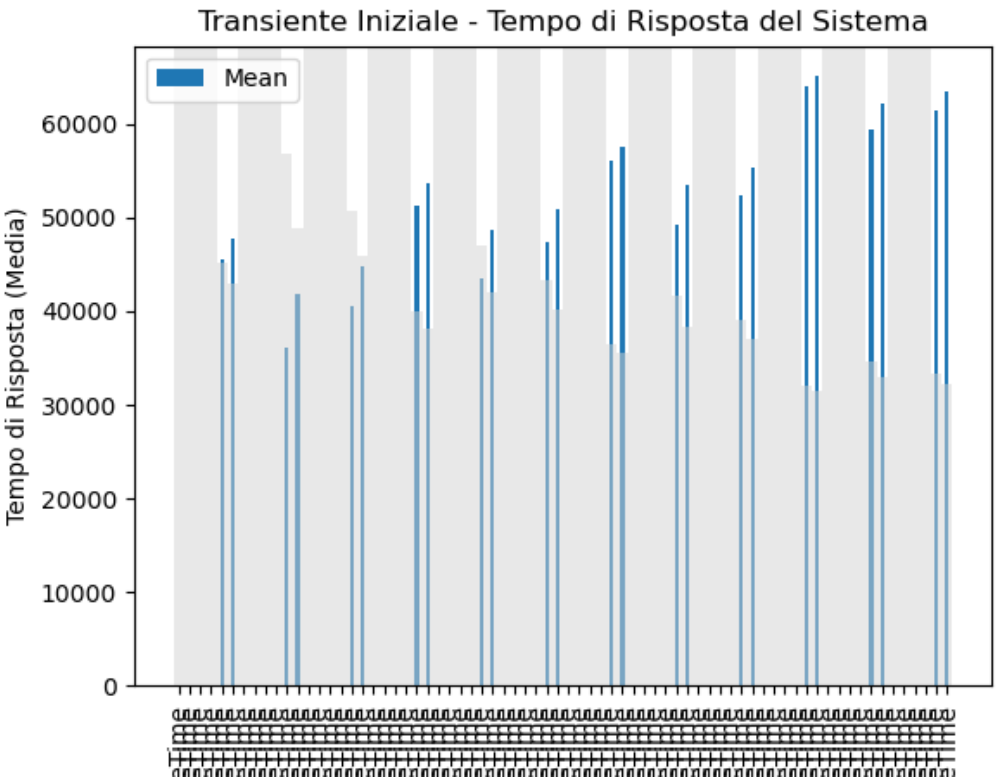

Figure 2: Transiente iniziale

Si è deciso dunque di eliminare questa fase e analizzare il sistema dal momento in cui 'e a pieno regime. La fase transiente individuata per questo sistema 'e di 30000 unità di tempo simulato, su un totale di 100000 unità di tempo della simulazione.

6 Risultati

La generazione dei risultati avviene attraverso il comando tramite il comando 'opp_scavetool', che elabora i risultati ottenuti durante le simulazioni in file CSV. I file CSV contengono i dati relativi ai diversi segnali di interesse monitorati durante le simulazioni. Per combinare i dati da ogni configurazione di simulazione in un unico file CSV finale, sono stati sviluppati opportuni script Python. Questi script analizzano i file CSV di ciascuna configurazione e estraggono le informazioni rilevanti per ciascun segnale di interesse. Successivamente, i dati vengono aggregati e salvati in un unico file CSV finale.
L'automazione di questo processo attraverso script Python consente di semplificare e velocizzare la creazione dei file CSV finali dei risultati. I dati ottenuti da ogni configurazione di simulazione vengono accuratamente combinati in un unico file, facilitando l'analisi e la visualizzazione dei risultati complessivi.
L'utilizzo degli script Python offre anche la flessibilità di personalizzare l'elaborazione dei dati in base alle specifiche esigenze dell'analisi. È possibile applicare calcoli statistici, calcolare medie, deviazioni standard o intervalli di confidenza .
In definitiva, l'automazione tramite script Python semplifica e ottimizza il processo di creazione dei file CSV finali dei risultati, consentendo un'analisi efficiente e una presentazione chiara dei dati ottenuti da ogni configurazione di simulazione:

- system_response_time
- server_response_time
- factor_utilization_data
- lifeTime_mean

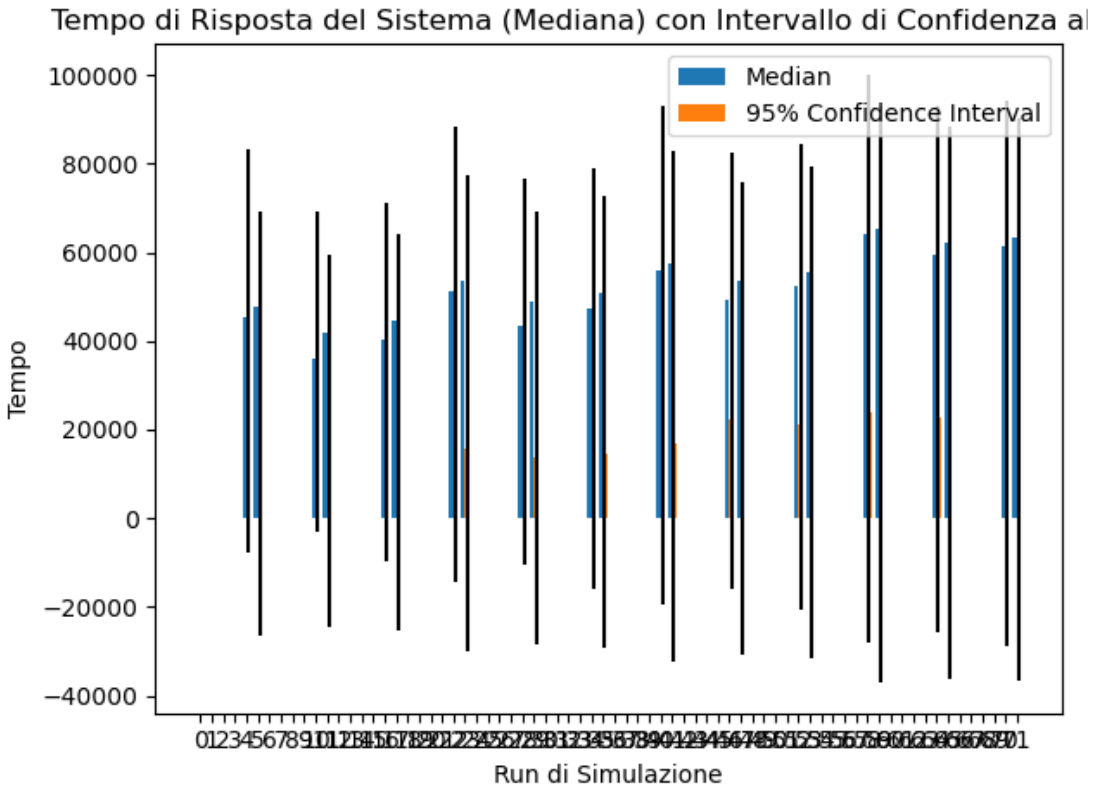

Figure 3: Tempo di Risposta del Sistema

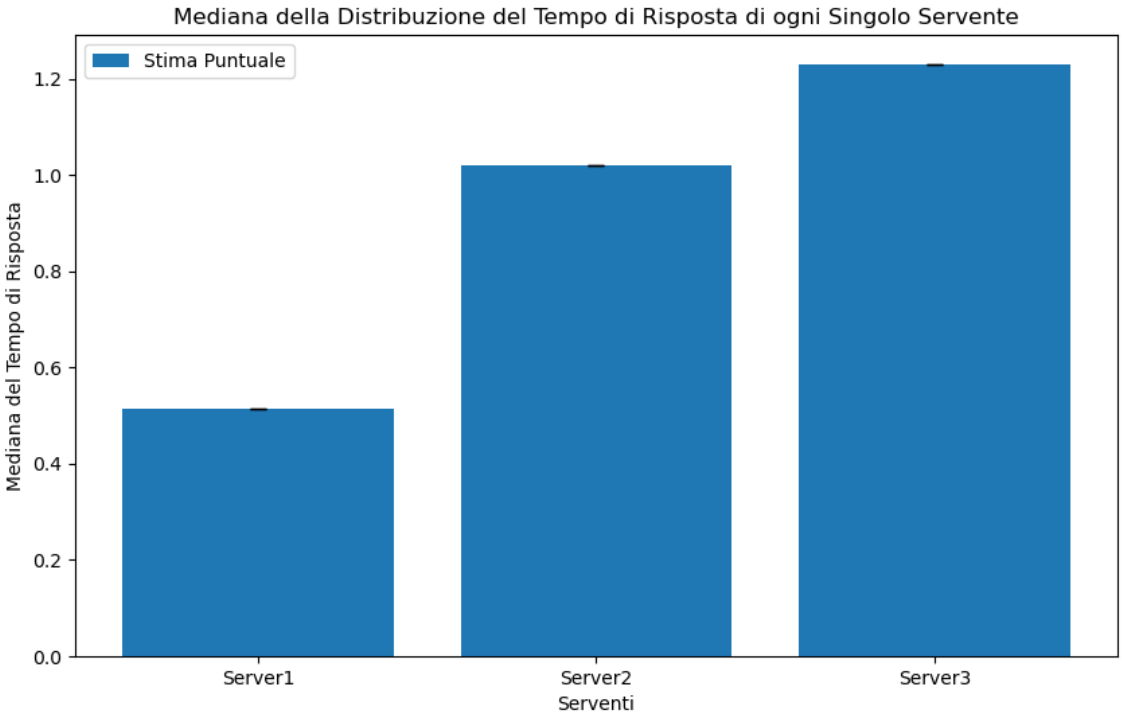

Figure 4: Tempo di Risposta dei Server

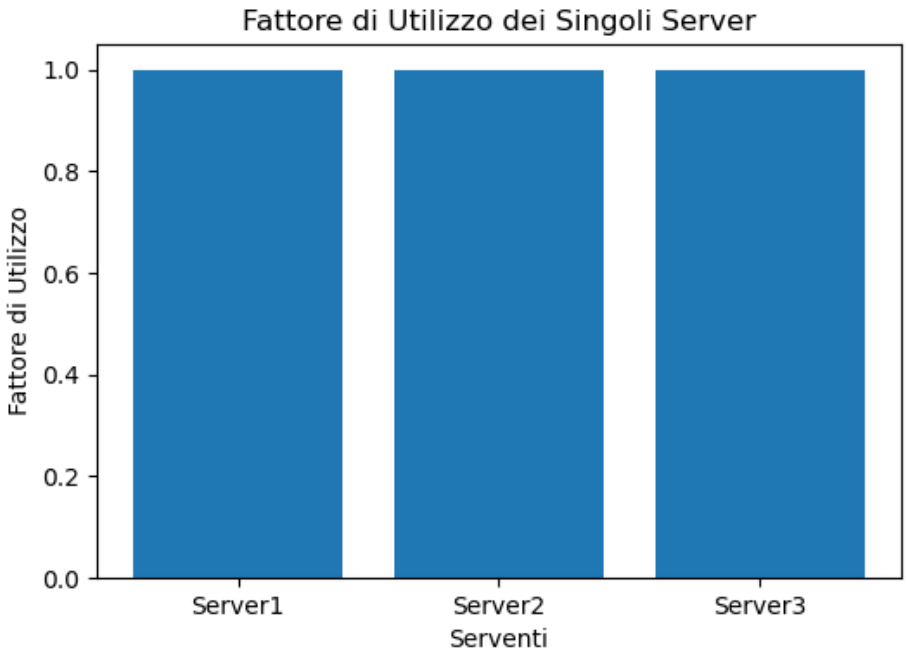

Figure 5: Fattore di Utilizzo dei Server

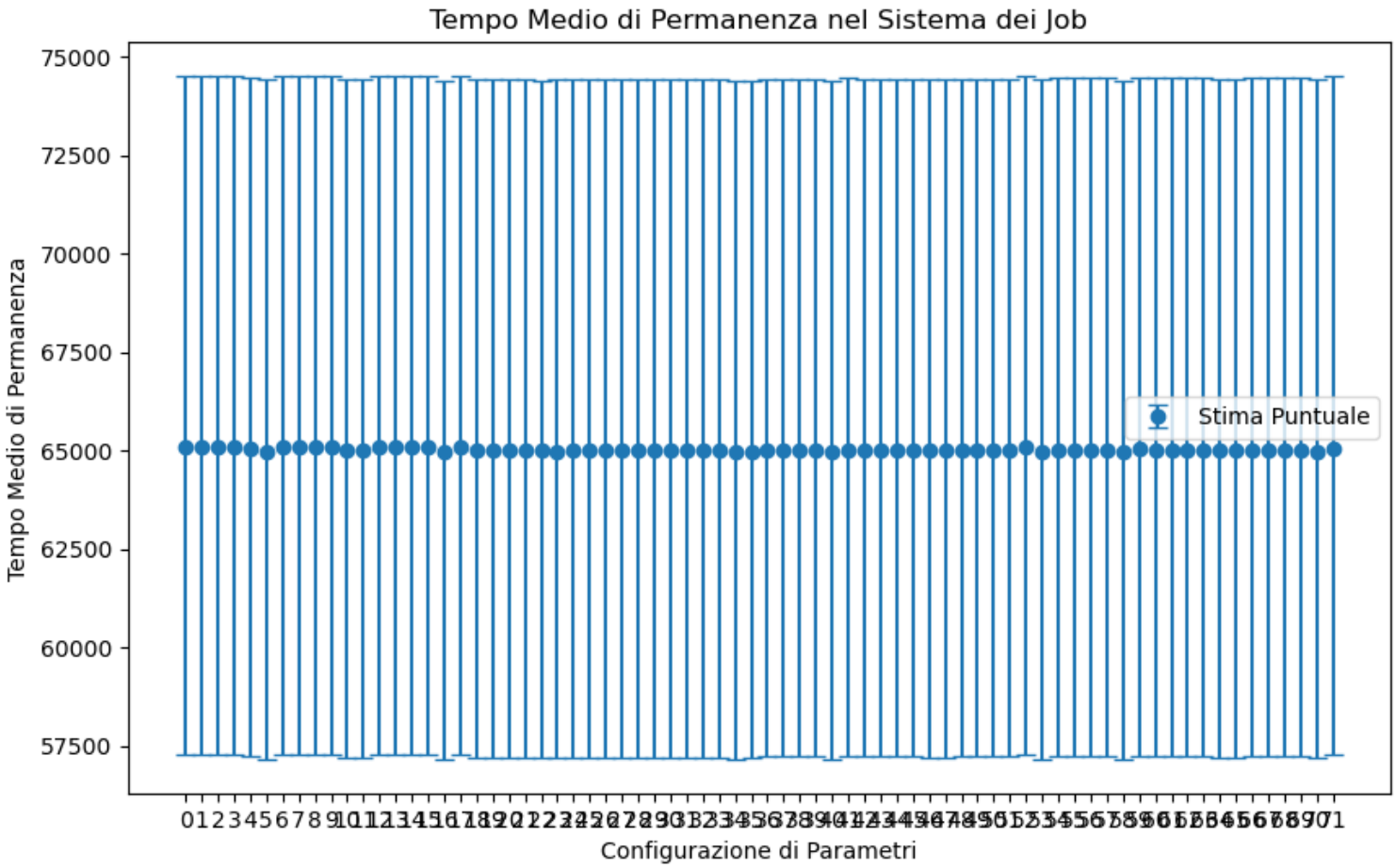

Figure 6: Tempo medio di Permanenza dei Job