Performance Modeling of Computer Systems and Networks
Relazione

Di Somma Alessia
Nedia Salvatore

Sommario

Nell'ambito del corso di Performance Modeling Of Computer Systems And Networks, anno accademico 2018-2019, abbiamo realizzato il progetto in oggetto con riferimento alla traccia assegnata.[1].Lo scopo del progetto è quello di analizzare un sistema a code calcolandone le performance sia dal punto di vista teorico, sia della simulazione.

[1] progettoGruppo9CFU1819b2.pdf

Indice

Modello concettuale

In riferimento alla traccia sono stati modellati e implementati due algoritmi.

1.1 Algoritmo 1

Nella modellazione dell' *algoritmo 1* è stato utilizzato un controllore che si occupa di decidere chi, tra cloud e cloudlet, debba servire un task. È stato assunto che un infinite server G/M/∞ (analizzato in sezione 9.0.1) gestisca gli arrivi al cloud, poiché questo server ha risorse illimitate, e un multiserver M/H_2/N/N (analizzato in sezione 9.0.2) senza coda gestisca gli arrivi al cloudlet. Gli arrivi sono di tipo esponenziale mentre i tempi di servizio, seguono una distribuzione esponenziale ed iperesponenziale(H_2) per cloud che per cloudlet rispettivamente. Il sistema non ha code e l'accoglienza di un task al cloudlet viene gestita dal controller come rappresentato in figura.

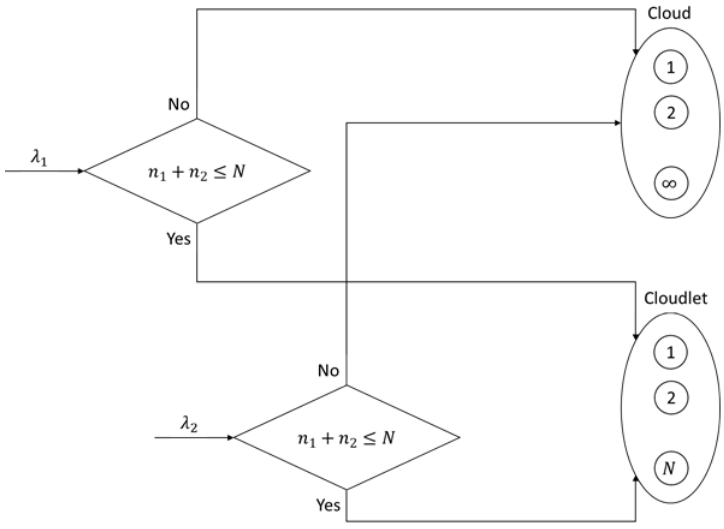


Figura 1.1: Modello della rete Algoritmo 1

Abbamo definito con:

- n_1: numero di task di classe 1 in esecuzione sul cloudlet
- n_2: numero di task di classe 2 in esecuzione sul cloudlet
- N: numero massimo di utenti nel cloudlet

1.2 Scelta dell'algoritmo 2

Per quanto riguarda l'*algoritmo 2* veniva richiesto di progettare un sistema con un controllo d'accesso differente, al fine di migliorare il tempo di risposta globale del sistema rispetto a quello ottenuto nell'*algoritmo 1*. È stato scelto perciò di utilizzare un sistema con prelazione, in cui si aggiunge la possibilità di migrare task di classe 2 dal cloudlet al cloud per favorire l'ingresso di uno di classe 1 se i task presenti nel cloudlet superano un certo valore di soglia *S*. Anche in questo caso, come in algoritmo 1 è stato assunto che un infinite server G/M/∞ (analizzato in sezione 9.0.1) gestisca gli arrivi al cloud, e un multiserver M/H_2/N/N (analizzato in sezione 9.0.2) senza coda gestisca gli arrivi al cloudlet. Nella fase di transizione dal cloudlet al cloud, per i task prelazionati, abbiamo tenuto conto dell'impossibilità di un job che viene spostato nel cloud di ripartire immediatamente. Per questo motivo abbiamo deciso di considerare anche un tempo di set up iniziale nel calcolo del tempo di risposta. Tale tempo di set up è stato settato a 0.5 secondi. I passi principali sono indicati nello pseudocodice:

```
Algoritmo 2;
Arrivi di classe 1;
if n1=N then
    send on cloud;
end
if n1+n2<S then
    accept task on cloudlet;
    if n2 > 0 then
        preemption;
    else
        accept task on cloudlet;
    end
end
Arrivi di classe 1;
if n1+n2≥S then
    send on cloud;
else
    accept task on cloudlet;
end
```

Di seguito, in figura, il modello di rete per l'*algoritmo 2*.

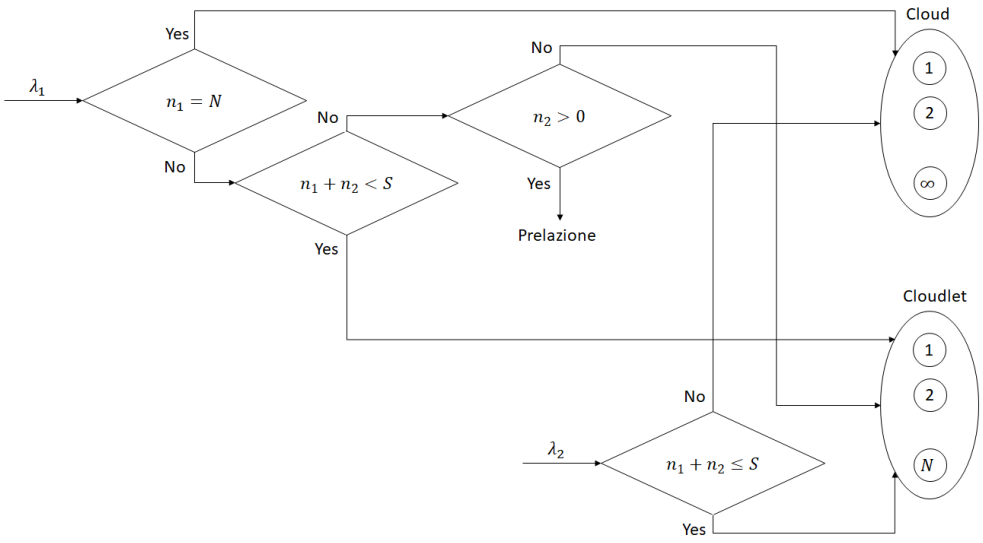


Figura 1.2: Modello della rete Algoritmo 2

In entrambi gli algoritmi la scelta dell'assegnazione del task al server del cloudlet è stata effettuata randomicamente al fine di distribuire il carico in maniera equa.

Modello delle specifiche

- $\lambda_1 = 4$ task/s ,tasso di arrivo di task di classe 1
- $\lambda_2 = 6.25$ task/s. tasso di arrivo di task di classe 2,
- $\mu_{1,cloud} = 0.25$ task/s tasso di servizio server cloud per task di classe 1
- $\mu_{2,cloud} = 0.22$ task/s, tasso di servizio server cloud per task di classe 2,
- $\mu_{1,clet} = 0.45$ task/s,tasso di servizio server cloudlet per task di classe 1
- $\mu_{2,clet} = 0.27$ task/s,tasso di servizio server cloudlet per task di tipo 2
- $a_i =$ tempo di arrivo di un task i
- $c_i =$ completamento di un job i
- $d_i = c_i - a_i$ durata di un task i
- $n_1 =$ numero di task di tipo 1
- $n_2 =$ numero di task di tipo 2
- $N = n_1 + n_2$ numero di task totali
- $t = 0$ tempo iniziale
- $n = 0$ per $t = 0$

Modello analitico

Dall'analisi concettuale, per il cloudlet, si evince che questo sia modellabile come un sistema di N serventi con dimensione di coda pari a zero. Come specificato nella traccia, per semplicità, nell'analisi del modello le distribuzioni dei tempi di servizio sono state considerate come se fossero di tipo esponenziale anzchè iperesponenziale; questo proprio per rendere semplificato il calcolo di tutte le stime e in particolare per premettere il confronto con i dati otenuti nella simulazione. Per ricavarne le statistiche medie necessarie a soddisfare i requisiti e gli obiettivi prefissati, il sistema è stato studiato come un processo markoviano. Ciò è stato possibile perché gli eventi di arrivo e di completamento, unici responsabili di transizioni di stato del sistema, sono di tipo esponenziale. Inoltre essendo il numero di stati discreto e finito, e, sapendo che la probabilità di transitare da un nodo al successivo dipende solo dall'istante in cui si trova il sistema e non dai passi precedenti, si è utilizzato il modello a catena di Markov per ricavare le probabilità di ciascuno stato.

Per quanto riguarda l'algoritmo 1 lo stato di ogni nodo della catena è definito solamente dal numero di task presenti all'interno del sistema, che siano di classe 1 o classe 2 indifferentemente. La catena di markov risutante è quella rappresentata in figura, in cui i tassi di servizio e di arrivo sono quelli forniti dalla traccia e indicati nel modello delle specifiche. Le transizioni, ovvero, i cambiamenti di stato avvengono invece solo se nel sistema si verificano eventi di arrivo o di completamento. I tassi di transizione sono dati dalle frequenze di arrivo o di completamento fornite nella traccia.
Una volta impostate le condizioni di bilanciamento dei flussi sui nodi, si ricavano le equazioni che hanno come incognite le probabilità limite di stato. Per generare le condizioni di bilanciamento dell'algoritmo 1 è stato utilizzato il metodo dei tagli in merito al fatto che la catena risultava essere abbastanza semplice, cosa che per l'algoritmo 2 non è stato possibile in quanto la catena risultava essere di dimensioni molto maggiori. Il sistema generale di bilanciamento dei flussi ha la seguente forma :

$$\begin{cases} \Pi_0(\lambda_1+\lambda_2) = \Pi_1(\mu_1+\mu_2) \\ \dots \\ \Pi_i(\lambda_1+\lambda_2) = \Pi_{i+1}(i+1)(\mu_1+\mu_2) \\ \dots \\ \Pi_{N-1}(\lambda_1+\lambda_2) = \Pi_N N(\mu_1+\mu_2) \\ \sum_{i=0}^{N} \Pi_i = 1 \end{cases} \Rightarrow \begin{cases} \Pi_0 = \frac{1}{\sum_{i=0}^{N} (\frac{\lambda}{\mu})^i (\frac{1}{i!})} \\ \dots \\ \Pi_i = \frac{\Pi_0(\lambda_1+\lambda_2)^i}{i!(\mu_1+\mu_2)^i} \\ \dots \\ \Pi_N = \frac{\Pi_0(\lambda_1+\lambda_2)^N}{N!(\mu_1+\mu_2)^N} \end{cases} \tag{3.1}$$

Nel nostro caso con N = 20:

$$\Pi_0 = \frac{1}{\sum_{i=0}^{20} (\frac{\lambda_1+\lambda_2}{\mu_1+\mu_2})^i (\frac{1}{i!})} \tag{3.2}$$

Risolte le equazioni si ottengono le probabilità teoriche cercate che forniscono un primo risultato analitico per il sistema. Attraverso esse è possibile ricavare le probabilità che il sistema sia vuoto (Π_0) e da questa la probabilità che il clouldlet sia pieno (Π_N), con cui il controller del sistema gestisce le diverse richieste e, di conseguenza, i relativi flussi di entrata dei diversi task.

In particolare:

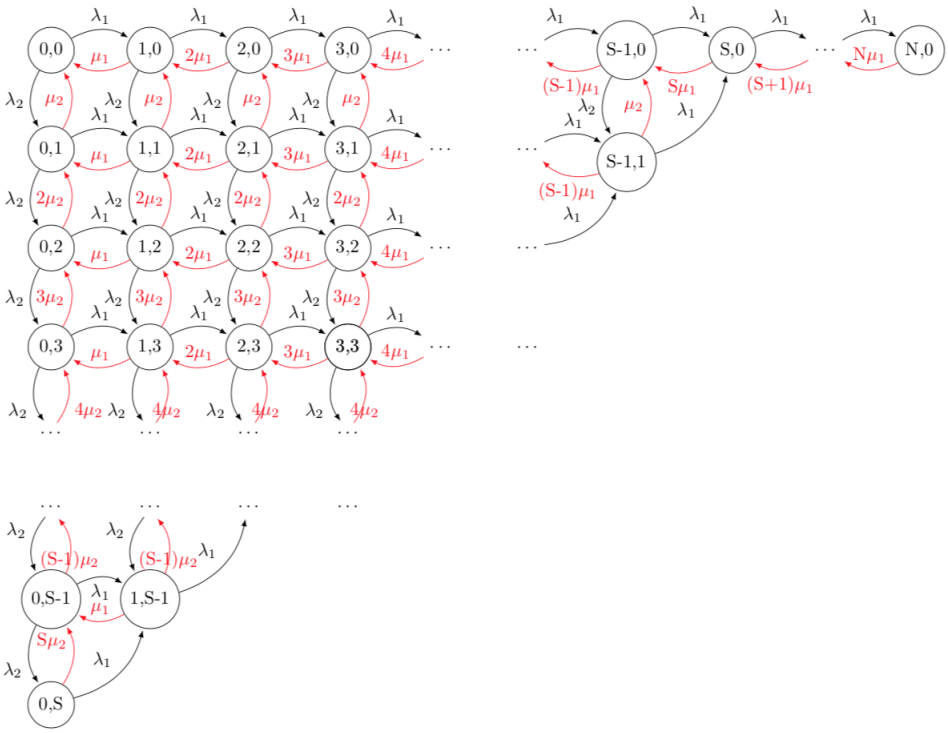
- $p_{cloud} = \Pi_{20} = 0.414549$ probabilità di un job di essere inviato al cloud
- $p_{clet} = 1 - \Pi_{20} = 0.585451$ probabilità di un job di essere inviato al cloudlet
- $global\_throughput = \lambda_1 + \lambda_2 = 10.25$ task/s throughput globale
- $throughput\_class1 = \lambda_1 = 4$ task/s throughput della classe 1
- $throughput\_class2 = \lambda_2 = 6.25$ task/s throughput della classe 2
- $throughput\_cloud = p_{cloud}(\lambda_1 + \lambda_2) = 4.24912725$ task/s throughput del cloud
- $throughput\_clet = p_{clet}(\lambda_1 + \lambda_2) = 6.00087275$ task/s throughput del cloudlet
- $throughput\_cloud\_class1 = p_{cloud}\lambda_1 = 1.658196$ task/s throughput del cloud per task di classe 1
- $throughput\_cloud\_class2 = p_{cloud}\lambda_2 = 2.59093125$ task/s throughput del cloud per task di classe 2
- $throughput\_clet\_class1 = p_{clet}\lambda_1 = 2.341804$ task/s throughput del cloudlet per task di classe 1
- $throughput\_clet\_class2 = p_{clet}\lambda_2 = 3.65906875$ task/s throughput del cloudlet per task di classe 2
- $E(T_{S1,clet}) = \frac{1}{u_{1,clet}} = 2.\overline{2}$ task/s tempo di risposta nel cloudlet per task di classe 1
- $E(T_{S2,clet}) = \frac{1}{\mu_{2,clet}} = 3.\overline{703}$ task/s tempo di risposta nel cloudlet per task di classe 2
- $E(T_{S1,cloud}) = \frac{1}{\mu_{1,cloud}} = 4$ task/s tempo di risposta nel cloud per task di classe 1
- $E(T_{S2,cloud}) = \frac{1}{\mu_{1,cloud}} = 4.\overline{54}$ task/s tempo di risposta nel cloud per task di classe 2

- $E(T_{Sclet}) = \frac{\lambda_1}{\lambda_1+\lambda_2}E(T_{S1,clet}) + \frac{\lambda_2}{\lambda_1+\lambda_2}E(T_{S2,clet}) = 3.125564$ task/s tempo di risposta nel cloudlet
- $E(T_{Scloud}) = \frac{\lambda_1}{\lambda_1+\lambda_2}E(T_{S1,cloud}) + \frac{\lambda_2}{\lambda_1+\lambda_2}E(T_{S2,cloud}) = 4.332594$ task/s tempo di risposta nel cloud
- $E(T_{S1}) = E(T_{S1,clet})(1 - \Pi_{20}) + E(T_{S1,cloud})\Pi_{20} = 2.959198$ task/s tempo di risposta per job di classe 1
- $E(T_{S2}) = E(T_{S2,clet})(1 - \Pi_{20}) + E(T_{S2,cloud})\Pi_{20} = 4.052650$ task/s tempo di risposta per job di classe 2
- $E(T_S) = \frac{\lambda_1}{\lambda_1+\lambda_2}E(T_{S1}) + \frac{\lambda_2}{\lambda_1+\lambda_2}E(T_{S2}) = 3.6259$ task/s tempo di risposta del sistema globale

Per quanto riguarda l' *algoritmo 2*, come già detto in precedenza, non è stato possibile risolvere a mano (senza l'aiuto di un software) le equazioni di bilanciamento. Questo in merito al fatto che, come si può vedere, la catena di Markov ottenuta (Figura 3.3) ha un numero di stati quadratico rispetto al valore di S, in particolare il valore esatto si ottiene dalla seguente relazione:

$$numero\_stati = N + 1 + \sum_{k=1}^{S} k = \frac{S(S+1)}{2} + N + 1 \tag{3.3}$$

Figura 3.3: Modello della catena di Markov per il cloudlet



Per quanto concerene l'equazioni di bilanciamento dei flussi, non potendo rappresentare tutto il sistema come nell' algoritmo 1 abbiamo deciso di utilizzare una equazione generale che racchiudesse tutte le equazioni, e che poi è anche quella che è stata utilizzata anche nello script in MatLab per risolverlo. L'equazione generale di bilanciamento dei flussi ha quindi la seguente forma generale: dove x e y sono le variabili indicanti lo stato del sistema, rispettivamente il numero di task di tipo 1 e il numero di task di tipo 2.

$$\Pi_{x,y}(a\lambda_1 + b\lambda_2 + y\mu_2 + x\mu_1) = \Pi_{x-1,y}(c\lambda_1) + \Pi_{x,y-1}(d\lambda_2) + \Pi_{x+1,y}(e(x+1)\mu_1) + \Pi_{x,y+1}(f(x+1)\mu_2) + \Pi_{x-1,y+1}(g\lambda_1) \tag{3.4}$$

I coefficienti *a,b,.....,g* si comportano come delle funzioni indicatrici. Questi assumono, quindi, i valori 1 o 0 a seconda che il valore della probabilità debba comparire o meno in una determinata equazione in dipendenza della posizione dello stato nella catena di Markov. Risolte le equazioni si ottengono le probabilità teoriche cercate che forniscono il risultato analitico per il sistema. Attraverso esse è possibile ricavare le probabilità con cui il controller del sistema gestisce le diverse richieste, inviandole o al cloudlet o al cloud, e, di conseguenza, i relativi flussi di entrata dei diversi task.
In particolare:

- $\Pi_{n_1+n_2<S} = \sum_{y=0}^{S-1} \sum_{x=0}^{S-y-1} \Pi_{x,y}$: probabilità che un task di seconda classe venga accettato nel cloudlet.
- $\Pi_{preemption} = \sum_{y=0}^{S-1} \sum_{x=S}^{S-y} \Pi_{x,y}$: probabilità che un task di seconda classe possa essere interrotto nel cloudlet.
- $p_{1,cloud} = \Pi_{N,0}$: probabilità di un job di essere inviato al cloud
- $p_{1,clet} = 1 - \Pi_{N,0}$: probabilità di un job di essere inviato al cloudlet
- $p_{2,cloud} = (1 - \Pi_{n_1+n_2<S})$: probabilità di un job di classe 2 di essere inviato al cloud
- $p_{2,clet} = \Pi_{n_1+n_2<S}$: probabilità di un job di classe 2 di essere accettato al cloudlet
- $p_{2,CletToCloud} = \frac{\lambda_{2,preemption}}{\lambda_{2,clet}}$: probabilità di un task di seconda classe di essere interrotto nel cloudlet
- $\lambda_{1,clet} = \lambda_1(1 - \Pi_N)$: tasso di arrivi di tipo 1 al cloudlet
- $\lambda_{1,cloud} = \lambda_1 p_{1,cloud}$: tasso di arrivi di tipo 1 al cloud
- $\lambda_{2,clet} = \lambda_2(1 - \Pi_N)$: tasso di arrivi di tipo 2 al cloudlet
- $\lambda_{2,cloud} = \lambda_2(1 - \Pi_{n_1+n_2<S}) + \lambda_1(p_{2,CletToCloud})$: tasso di arrivi di tipo 2 al cloud
- $\lambda_{2,preemption} = \lambda_1 \Pi_{preemption}$: tasso di task di tipo due prelazionati
- $global\_throughput = \lambda_1 + \lambda_2 = 10.25$ task/s throughput globale
- $E(T_{S1,clet}) = \frac{1}{\mu_{1,clet}} = 2.\overline{2}$ task/s tempo di risposta nel cloudlet per task di classe 1
- $E(T_{S2,clet}) = \frac{1}{\mu_{2,clet}} = 3.\overline{703}$ task/s tempo di risposta nel cloudlet per task di classe 2
- $E(T_{S1,cloud}) = \frac{1}{\mu_{1,cloud}} = 4$ task/s tempo di risposta nel cloud per task di classe 1
- $E(T_{S2,cloud}) = \frac{1}{\mu_{1,cloud}} = 4.\overline{54}$ task/s tempo di risposta nel cloud per task di classe 2
- $E(T_{S1}) = p_{1,cloud}E(T_{S1,cloud}) + p_{1,clet}E(T_{S1,clet})$ tempo di risposta per job di classe 1
- $E(T_{S2}) = p_{2,cloud}E(S_{2,cloud}) + p_{2,clet} * (1 - p_{2,CletToCloud})E(T_{S2,clet}) + p_{2,clet} * p_{2,CletToCloud}E(T_{S2,cloud} + SETUP)$ tempo di risposta per job di classe 2
- $E(T_S) = \frac{\lambda_1}{\lambda_1+\lambda_2}E(T_{S1}) + \frac{\lambda_2}{\lambda_1+\lambda_2}E(T_{S2})$ tempo di risposta del sistema globale

Modello computazionale

4.1 Esecuzione del simulatore

Per poter utilizzare il simulatore è necessario disporre di una macchina avente il compilatore gcc e lanciare il makefile da linea di comando con il comando "make". Per semplicità e chiarezza sono stati creati due file separati per i due algoritmi (*sca1.c* e *sca2.c* rispettivamente). Non sono necessari parametri da passare come input al programma, tuttavia per poter modificare i parametri necessari è sufficente andare a modificare tali valori nelle MACRO presenti all'inizo del file *sca1*/sca2 a seconda che si tratti dell'*algoritmo 1* o dell'*algoritmo2*.

4.2 Generatore di matrice di transizione di stato

Il software che abbiamo scritto in Matlab consente di generare la matrice di transizione di stato della catena di Markov al variare dei parametri S e N. A questo punto ne verifica l'ergodicità ed irriducibilità usando l'Econometrics Toolbox di Matlab. Per provare l'ergodicità Matlab si avvale del teorema di Wielandt: una catena di Markov è ergodica se e solo se tutti gli elementi di P^m sono positivi per $m = (n-1)^2 + 1$, dove P è la matrice di transizione e n il numero degli stati del sistema. Il sistema in esame risulta essere ergodico per ogni valore di S compreso tra 1 ed N, al variare di N tra 1 e 24.

4.3 Simulazione Next-Event

Il software che abbiamo sviluppato per simulare il sistema segue la logica Next-Event. Lo stato del sistema è definito come il numero di tasks presenti ad ogni istante, differenziato per tipologia e per centro. Ad ogni iterazione viene considerato il minimo valore degli eventi generati di arrivo o completamento e ne viene valutato l'impatto sullo stato del sistema. Se si tratta di un arrivo, viene generato il successivo ed eseguita la logica del controller per gestire il task.

4.4 Risolutore probabilistico del sistema

Il software scritto in Matlab consiste in diverse funzioni che agiscono a livello probabilistico. Il flusso principale di esecuzione di questi script è formato dalla generazione, in primis, della catena di Markov e delle sue equazioni. A questo punto, lo script, le risolve e computa

tutte le probabilità necessarie per determinare i throughput, i tempi di risposta e il numero medio di task nel sistema. Questi valori teorici vengono poi esportati in un file e presi come riferimento nel momento in cui si graficano quelli restituiti dal simulatore del sistema.

Verifica e Validazione

5.1 Introduzione

Il processo di Verifica e Validazione è stato effettuato principalmente per l'algoritmo 2, data la sua complessità. Si è svolto verificando e validando prima il generatore e risolutore di equazioni relative alla catena di Markov scritta in Matlab che modella il cloudlet e, in seguito, facendo altrettanto sul software di simulazione del sistema in sé.

5.2 Generatore di matrici di transizione di stato

Per validare il generatore di matrici di transizione di stato per la catena di Markov abbiamo generato una matrice di dimensione $n \times n$ dove n è calcolato tramite l'equazione 3.3. Abbiamo verificato che il risultato ottenuto fosse effettivamente il risultato che ci attendevamo, calcolandolo senza l'aiuto del software. Abbiamo condotto due test per la validazione con valori di N ed S abbastanza piccoli da poter effettuare i calcoli. In particolare abbiamo ottenuto i seguenti risultati rispettivamente per N=1, S=1 e N=5, S=3 (Tabelle 5.2 5.1).

Tabella 5.1: Matrice di transizione di stato per N = 5, S = 3

	Π_{00}	Π_{10}	Π_{20}	Π_{30}	Π_{40}	Π_{50}	Π_{01}	Π_{11}	Π_{21}	Π_{02}	Π_{12}	Π_{03}
Π_{00}	0	λ_1	0	0	0	0	λ_2	0	0	0	0	0
Π_{10}	μ_1	0	λ_1	0	0	0	0	λ_2	0	0	0	0
Π_{20}	0	$2\mu_1$	0	λ_1	0	0	0	0	λ_2	0	0	0
Π_{30}	0	0	$3\mu_1$	0	λ_1	0	0	0	0	0	0	0
Π_{40}	0	0	0	$4\mu_1$	0	λ_1	0	0	0	0	0	0
Π_{50}	0	0	0	0	$5\mu_1$	0	0	0	0	0	0	0
Π_{01}	μ_2	0	0	0	0	0	0	λ_1	0	λ_2	0	0
Π_{11}	0	μ_2	0	0	0	0	μ_1	0	λ_1	0	λ_2	0
Π_{21}	0	0	μ_2	λ_1	0	0	0	$2\mu_1$	0	0	0	0
Π_{02}	0	0	0	0	0	0	$2\mu_2$	0	0	0	λ_1	λ_2
Π_{12}	0	0	0	0	0	0	0	$2\mu_2$	λ_1	μ_1	0	0
Π_{03}	0	0	0	0	0	0	0	0	0	$3\mu_2$	λ_1	0

Tabella 5.2: Matrice di transizione di stato per N = 1, S = 1

	p_{00}	p_{10}	p_{01}
p_{00}	0	λ_1	λ_2
p_{10}	μ_1	0	0
p_{01}	μ_2	λ_1	0

5.3 Generatore e risolutore di equazioni

Per quanto riguarda il simulatore di equazioni, abbiamo verificato che il suo algoritmo rispettasse le specifiche che abbiamo posto durante il design dello stesso. Questo doveva prevedere la generazione di una matrice $N \times S$ per ricreare virtualmente la conformazione degli stati della catena di Markov che, in questo caso, si dispone bene bidimensionalmente. In aggiunta doveva poter generare un'equazione di bilanciamento per ogni stato della catena, di cui una è linearmente dipendente nel sistema, e la somma di tutte le probabilità pari ad uno (condizione di normalizzazione). Il sistema generato doveva essere poi risolto, assegnando ad ogni probabilità incognita un valore compreso tra 0 ed 1.
Dopodiché abbiamo proceduto alla validazione del software. Per avere una buona fiducia nei confronti del prodotto, abbiamo confrontato i valori ottenuti dal software con quelli da noi calcolati, per valori sufficientemente bassi di S ed N. A causa del valore quadratico con cui cresce il numero di stati della catena ($O(S^2)$) non è stato possibile, per motivi di praticità, testare ulteriormente il software.

5.4 Simulatore del sistema

Il simulatore del sistema ha dapprima passato un test di verifica del suo codice in cui ci si è accertati che le sue specifiche fossero state rispettate durante la programmazione:

- Il Next-Event simulator doveva constare di uno stato semplicemente definito dal numero di task di tipo 1 e 2 nel cloudlet e nel cloud
- L'evento doveva essere definito da un arrivo di un task di tipo 1 o 2 o un completamento di uno dei task serviti da uno dei server del cloudlet o del cloud.
- La generazione di valori casuali doveva essere effettuata tramite una libreria che permettesse l'utilizzo del multistream.

Infine c'era necessità di tenere traccia dei dati aggregati per cloud, per cloudlet e per tipo di task e la possibilità di cambiare tutti i parametri della simulazione. Una volta verificata l'aderenza del codice C scritto alle specifiche, ne abbiamo testato la validità eseguendo più misurazioni per algoritmo.

Per quanto riguarda *l'argoritmo 1*, avendo il valore di N fisso a 20, abbiamo eseguito una prima prova con tale valore e successivamente abbiamo verificato che, abbassando ed aumentando il valore di N rispettivamente si ottengono tempi di risposta minori e maggiori ai precenti.

Tabella 5.3: Tempi di risposta simulati per N = 20

	Valori teorici	Valori simulati
Tempo risposta globale (s)	3.625938	3.622958
Tempo risposta classe 1 (s)	2.959198	2.970106
Tempo risposta classe 2 (s)	4.05265	4.041308
Tempo risposta clet (s)	3.125564	3.120641
Tempo risposta cloud (s)	4.332594	4.334284
Throughput globale (task/s)	10.25	10.249711
Throughput classe 1 (task/s)	4	4.003705
Throughput classe 2 (task/s)	6.25	6.246006

Tabella 5.4: Confronto tra tempi di risposta simulati al variare di N

	N = 20	N = 10	N = 30
Tempo risposta globale (s)	3.622958	3.96959	3.324879
Tempo risposta classe 1 (s)	2.970106	3.466061	2.514346
Tempo risposta classe 2 (s)	4.041308	4.292238	3.844263
Tempo risposta clet (s)	3.120641	3.120741	3.120691
Tempo risposta cloud (s)	4.334284	4.333172	4.341897

Per quanto riguarda *l'argoritmo 2*, avendo il valore di N fisso a 20, abbiamo eseguito più prove variando il livello di soglia S ed abbiamo verificato che, ad un maggiore valore della soglia corrisponde un minore tempo di risposta e viceversa ad un minore valore della stessa corrisponde un maggiore tempo di risposta.

Tabella 5.5: Tempi di risposta simulati per N = 20, S = 20

	Valori teorici	Valori simulati
Tempo risposta globale (s)	3.496811	3.519627
Tempo risposta classe 1 (s)	2.231252	2.222095
Tempo risposta classe 2 (s)	4.509257	4.196501
Throughput globale (task/s)	10.25	10.231508
Throughput classe 1 (task/s)	4	3.989881
Throughput classe 2 (task/s)	6.25	6.241627

Tabella 5.6: Confronto tra tempi di risposta simulati al variare di S

	S = 5	S = 13	S = 20
Tempo risposta globale (s)	3.626594	3.528597	3.458544
Tempo risposta classe 1 (s)	2.227146	2.222457	2.222094
Tempo risposta classe 2 (s)	4.523296	4.362343	4.103553
Tempo risposta clet (s)	2.257335	2.596804	2.965695
Tempo risposta cloud (s)	4.536076	4.543723	4.733370

Analisi sui dati

Per verificare se il sistema fosse di titpo stazionario o transiente abbiamo deciso di analizzare i valori in entrambi i casi, per poi arrivare alla conclusione che il sistema risluta essere si tipo stazionario.
Le statistiche sono state effettuate considerando i modelli G/M/∞ per il cloud e M/H_2/N/N per il cloudlet.

6.1 Analisi nel transiente

Per quanto riguarda l'analisi nel transitorio abbiamo deciso di utilizzare il metodo della replicazione, effettuando vari run per valori del tempo crescente, sia per l'algoritmo 1 che per l'algoritmo 2. Siamo immediadamente giunti alla conclusione che si trattasse di un sitema di tipo stazionario. Questo andava proprio a rafforzare le nostre aspettative considerando il fatto che il cloud ha capacità infinita.
I valori ottenuti sono quelli indicati in tabella e, come si può facilmente notare, già introno al tempo $t = 10000$ secondi, i valori iniziano a convergere. In particolare sono stati effettutai cinque run con cinque seed differenti prenedendo come statistiche i valori medi del tempo di risposta globale e del throughput. Questo perché tali parametri ci sembravano essere valori abbastanza rappresentatvi dell'andamento del sistema.

Tabella 6.7: Valori del tempo di risposta globale nell'algoritmo 1 al variare di t

Seed	123456	1346166132	676287059	455285772	90542590
t = 10 (s)	3.664446	4.390200	432778	3.202285	3.503842
t = 50 (s)	3.723101	3.665332	3.739600	3.636100	3.461143
t = 100 (s)	3.717688	3.457376	3.745703	3.640934	3.608198
t = 500 (s)	3.595817	3.607881	3.690955	3.596632	3.683059
t = 1000 (s)	3.546524	3.688730	3.617446	3.558487	3.696890
t = 5000 (s)	3.615753	4.377898	3.621178	3.597749	3.630129
t = 10000 (s)	3.629972	3.625063	3.618333	3.601139	3.634721
t = 20000 (s)	3.629596	3.635329	3.627402	3.608226	3.631245
t = 25000 (s)	3.631481	3.639744	3.624890	3.606735	3.630344
t = 30000 (s)	3.622595	3.632854	3.629804	3.611493	3.633614

Tabella 6.8: Valori del throughput globale nell'algoritmo 1 al variare di t

Seed	123456	1346166132	676287059	455285772	90542590
t = 10 (s)	4.086496	5.628257	4.759103	5.279841	5.771961
t = 50 (s)	7.743422	8.452570	8.246072	8.279668	8.153299
t = 100 (s)	8.655600	9.189947	9.063025	9.301095	8.922451
t = 500 (s)	9.649848	10.136486	10.007715	10.133909	9.784825
t = 1000 (s)	9.877110	10.255420	10.141712	10.305748	9.958613
t = 5000 (s)	10.131733	10.246325	10.255785	10.341959	10.175544
t = 10000 (s)	10.199995	10.248931	10.261098	10.317878	10.222113
t = 20000 (s)	10.233519	10.249322	10.261526	10.299189	10.241120
t = 25000 (s)	10.238007	10.249243	10.261008	10.292200	10.246327
t = 30000 (s)	10.240429	10.249940	10.260005	10.289102	10.249153

Tabella 6.9: Valori del tempo di risposta globale nell'algoritmo 2 al variare di t

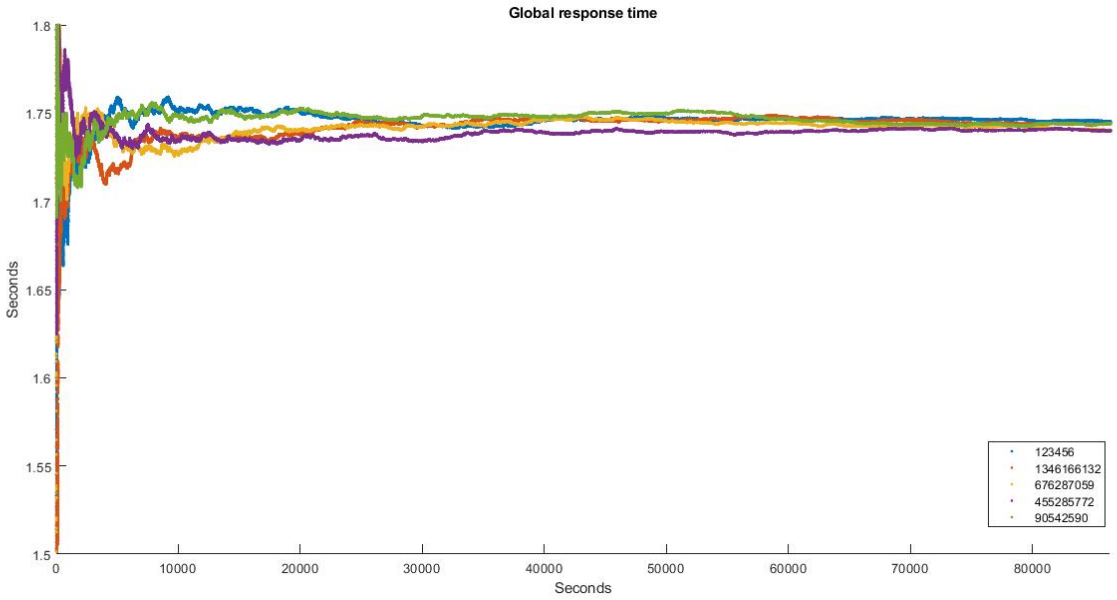
Seed	123456	1346166132	676287059	455285772	90542590
t = 10 (s)	1.320778	1.638442	1.960451	1.634185	2.032723
t = 50 (s)	1.502205	1.591021	1.923073	1.591787	1.652797
t = 100 (s)	1.572829	1.598397	1.806729	1.769758	1.739462
t = 500 (s)	1.652642	1.674689	1.725652	1.752253	1.739138
t = 1000 (s)	1.696857	1.704260	1.704151	1.745770	1.724056
t = 5000 (s)	1.733255	1.707775	1.735244	1.731340	1.738128
t = 10000 (s)	1.736704	1.730097	1.730801	1.731833	1.730350
t = 20000 (s)	1.730865	1.735995	1.736016	1.729286	1.729311
t = 25000 (s)	1.730298	1.741390	1.735147	1.731250	1.732856
t = 30000 (s)	1.730599	1.736505	1.733916	1.732715	1.737330

Tabella 6.10: Valori del throughput globale nell'algoritmo 2 al variare di t

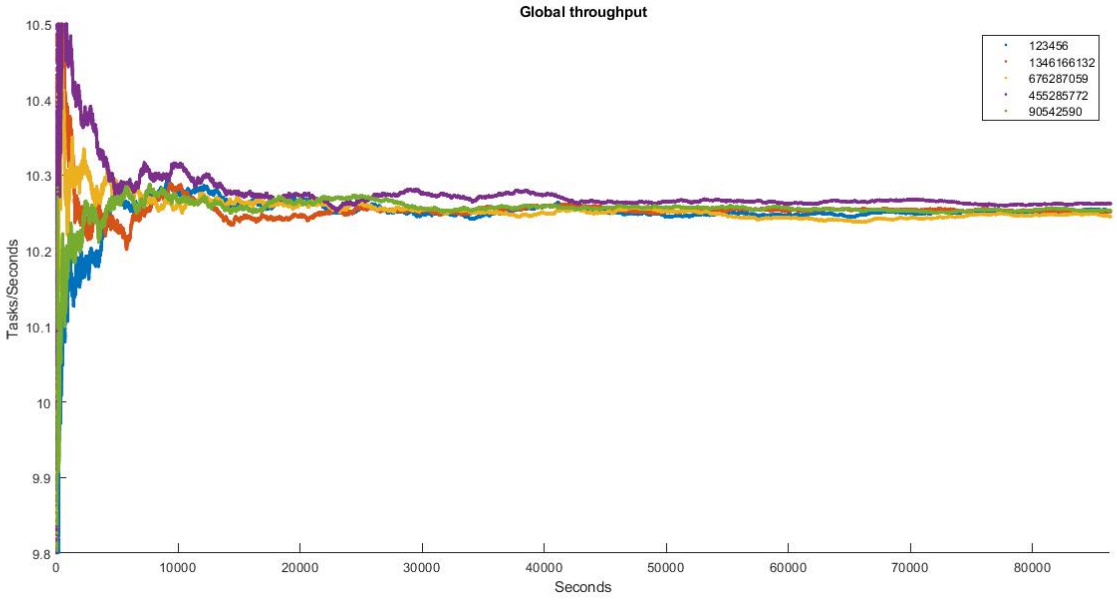
Seed	123456	1346166132	676287059	455285772	90542590
t = 10 (s)	5.768018	7.517732	6.430013	6.657667	6.606866
t = 50 (s)	8.660369	9.472813	9.162364	9.143130	8.854922
t = 100 (s)	9.256512	9.807639	9.654776	9.862705	9.397336
t = 500 (s)	9.824849	10.323184	10.191917	10.307749	9.940085
t = 1000 (s)	9.976086	10.364635	10.248182	10.406100	10.050349
t = 5000 (s)	10.157561	10.274662	10.283306	10.368610	10.199930
t = 10000 (s)	10.214255	10.264357	10.276233	10.332519	10.235667
t = 20000 (s)	10.241333	10.257735	10.269770	10.307206	10.248573
t = 25000 (s)	10.244440	10.256142	10.267774	10.298787	10.252461
t = 30000 (s)	10.245908	10.255809	10.265768	10.294706	10.254388

6.2 Analisi nello stazionario

Anche per quanto riguarda l'analisi nello stazionario abbiamo optato per l'esecuzione di cinque run piuttosto lunghi che ci permettessero di constatare in maniera visiva se i valori medi del throughput e del tempo di risposta globale convergessero o meno.
Abbiamo eseguito la simulazione per entrambi gli algoritmi scegliendo un tempo di simulazione pari a 86400 secondi e, per l'algoritmo 2, abbiamo scelto S uguale a 20. Ogni risultato è stato calcolato da una replica indipendente del run di simulazione (ottenuta partendo da una generazione dei numeri casuali con diverso seed) sia per i tempi di risposta che per i throughput globali. I seed ultilizzati sono rispettivamente: 123456 (blu), 1346166132 (rosso), 676287059 (giallo), 455285772 (viola) e 90542590 (verde).
Come è possibile vedere dai plot, i risultati sembrano suggerirci in maniera piuttosto chiara, sia per l'algoritmo 1 che per l'algoritmo 2, che il sistema tende alla stazionarietà. I risultati rispecchiano ciò che effettivamente ci aspettavamo da questo sistema.

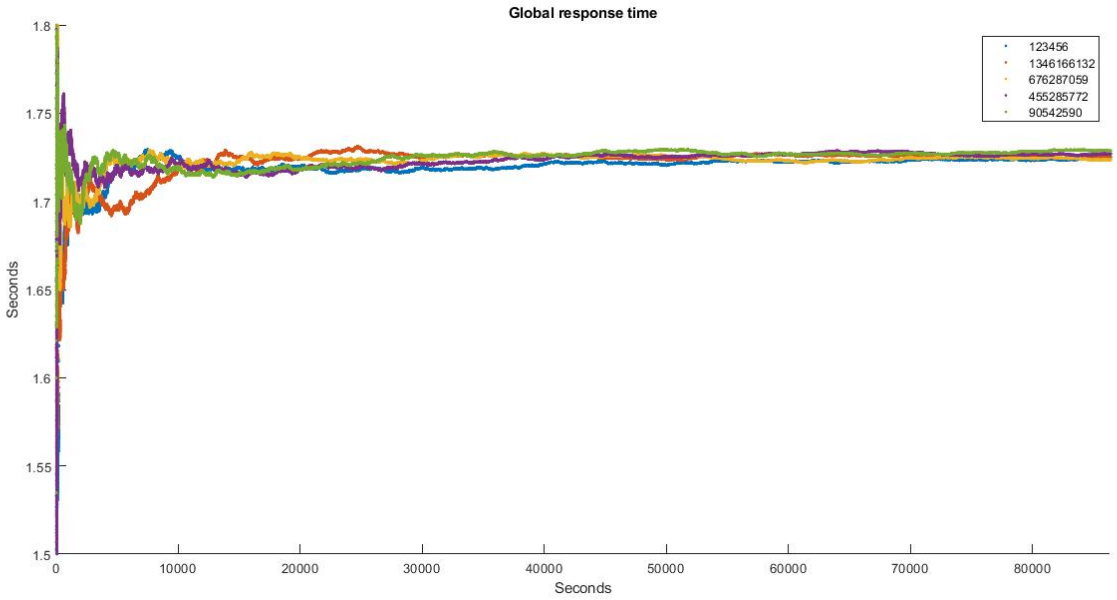


(a) Tempo di risposta globale (algoritmo 1)

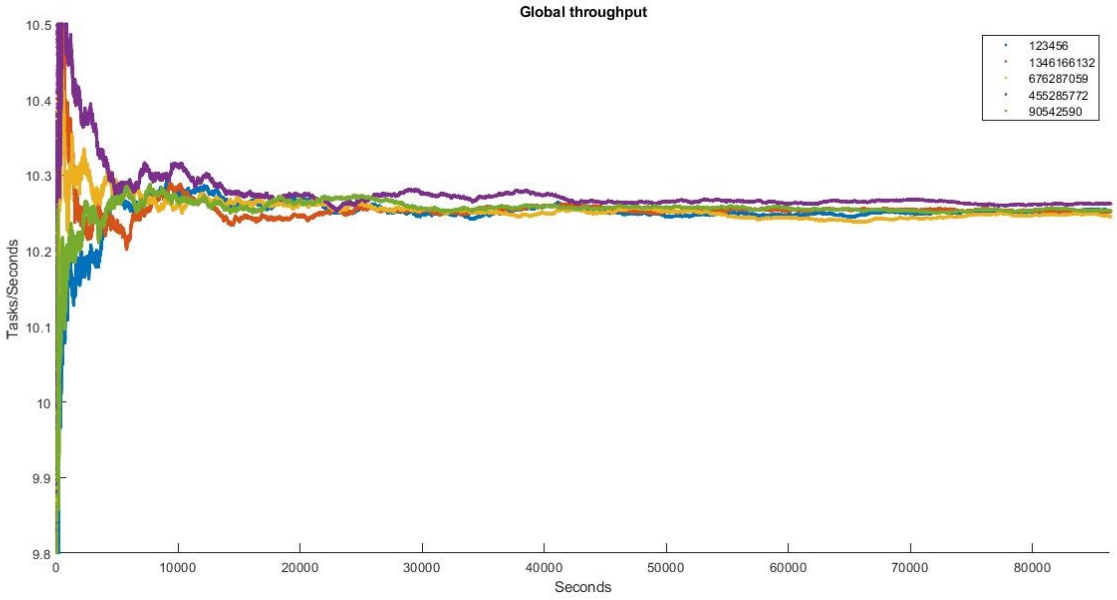


(b) Throughput globale (algoritmo 1)

Figura 6.4: Statistiche a regime per l'algoritmo 1



(a) Tempo di risposta globale (algoritmo 2)



(b) Throughput globale (algoritmo 2)

Figura 6.5: Statistiche a regime per l'algoritmo 2

Metodologie utilizzate

7.1 Batch means

Per ottenere le statistiche a regime richieste non è stato usato l'approccio di replicazione spiegato nella sezione precedente che risente, in ogni run, del bias iniziale essendo il sistema vuoto al momento dell'inizializzazione.
Per ottenere i dati necessari al calcolo dei valori medi si è utilizzato il metodo dei batch means. È stato eseguito un run sufficientemente lungo, di 86400 secondi, che è stato diviso in n batch di lunghezza fissa di 3600 secondi. Per il calcolo delle popolazioni medie, per ciascuna partizione sono state calcolate le statistiche mediandole sul tempo trascorso:

$$\overline{x} = \frac{1}{t_n} \sum_{i=1}^{n} x_i \delta_i \qquad (\delta_i = t_i - t_{i-1}) \tag{7.5}$$

E mediandole su ogni misurazione per il calcolo delle altre metriche:

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{7.6}$$

Sfruttando l'algoritmo one-pass di Welford queste sono state computate a runtime permettendo il raffinamento dei dati senza il salvataggio in memoria degli stessi in formato grezzo.
Per le statistiche temporali:

$$\overline{x}_i = \overline{x}_{i-1} + \frac{\delta_i}{t_i}(x_i - \overline{x}_{i-1}) \tag{7.7}$$

$$v_i = v_{i-1} + \frac{\delta_i t_{i-1}}{t_i}(x_i - \overline{x}_{i-1})^2 \tag{7.8}$$

Mentre per le rimanenti:

$$\overline{x}_i = \overline{x}_{i-1} + \frac{1}{i}(x_i - \overline{x}_{i-1}) \tag{7.9}$$

$$v_i = v_{i-1} + \frac{i-1}{i}(x_i - \overline{x}_{i-1})^2 \tag{7.10}$$

Al termine di ciascun batch le statistiche parziali sono state azzerate a differenza dello stato del sistema, così da permettere al successivo di ricalcolare i valori da zero.
La scelta della durata della simulazione e di ciascun batch è stata effettuata secondo le seguenti linee guida:

- Il numero di batch dovrebbe essere tra b = 10 e b = 30 (Schmeiser 1982).
- La lunghezza di ciascun batch dovrebbe essere scelta affinché il modulo dell'autocorrelazione con lag pari a uno tra le medie dei batch sia minore di 0.2 (Banks, Carson, Nelson, Nicol 2001).

Nel nostro caso il numero di batch utilizzato è pari a 24.

Statistiche a regime

Assunta la stazionarietà del sistema, ne sono state studiate le metriche a regime, eseguendo la simulazione per un periodo sufficientemente grande. Inoltre per validare la simulazione sono stati confrontati i valori delle medie simulate con i valori teorici, trovati sfruttando il modello analitico, ipotizzando un tempo di servizio esponenziale per entrambi i server (cloudlet e cloud), come già specificato nelle sezioni precedenti. Tutti gli intervalli hanno una confidenza del 95%.

8.1 Statistiche per l'algoritmo 1

Parametri di simulazione

- Tempo totale: 86400 secondi
- Seed iniziale: 123456
- Dimensione del batch: 3600 secondi

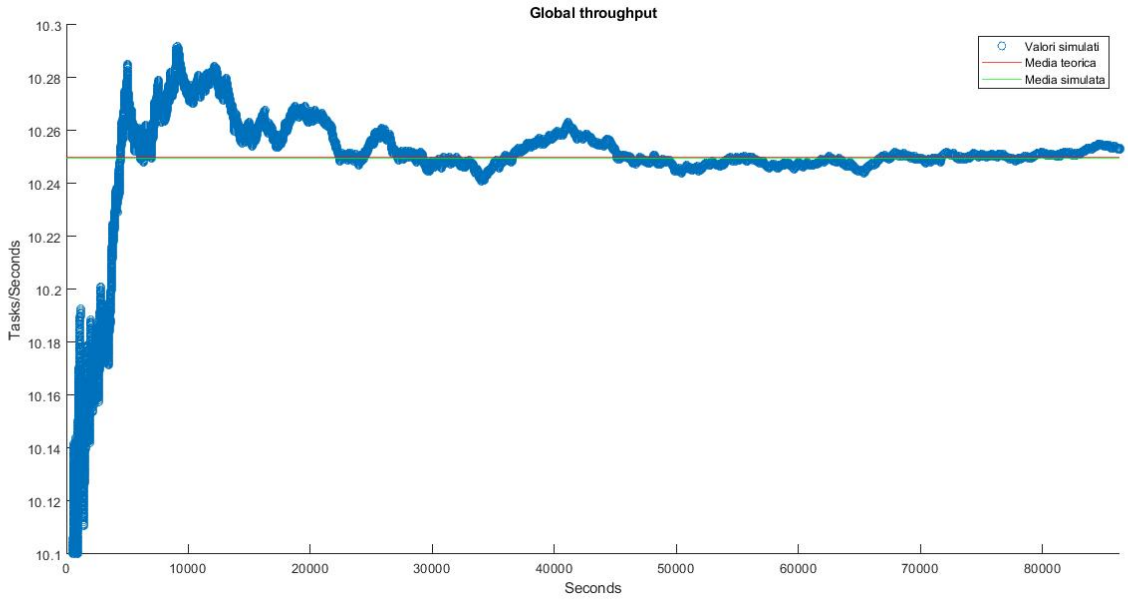
8.1.1 Tempo di risposta e throughput globale e per classe

Throughput

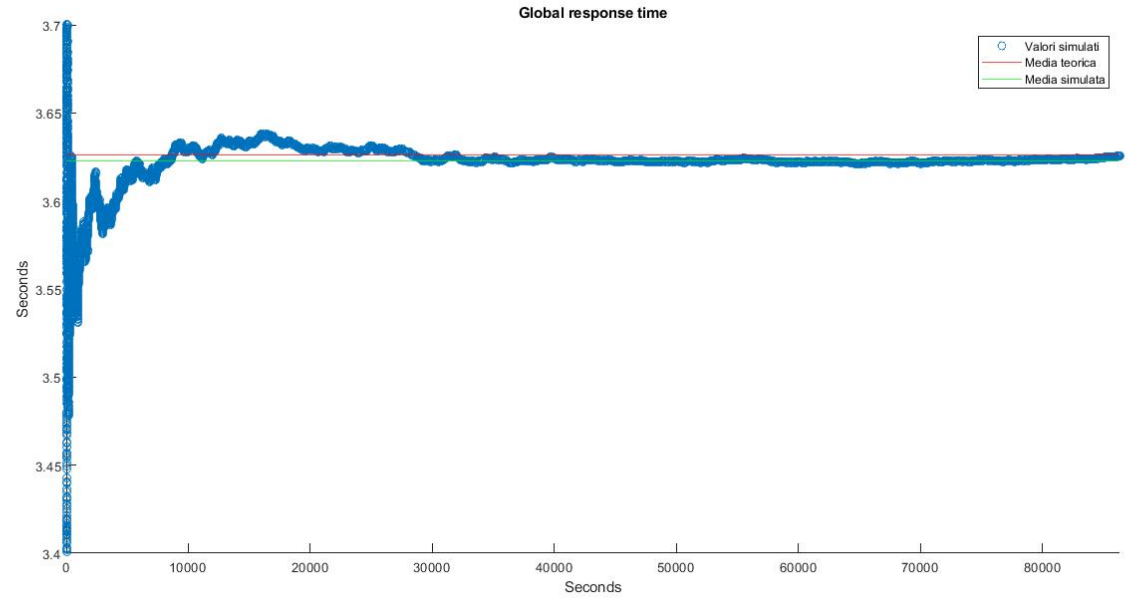
Per quanto riguarda il throughput globale risulta essere dato dalla somma dei throughput di classe 1 e classe 2 come atteso dal modello analitico. Tutti i throughput tendono a crescere in quanto il sistema si riempie e tende alla stazionarietà, pertanto, il numero di task che vengono completati è basso in un primo momento e si stabilizza dopo un determinato intervallo di tempo. Tutti i grafici inoltre convergono alla media teorica.

Tempo di risposta

Per quanto riguarda i tempi di risposta, anche in questo caso i valori simulati tendono ai valori ottenuti dal modello analitico. In particolare, a differenza del throughput, i tempi di risposta si stabilizzano, più nei casi delle singole classi che nel caso globale, dopo un iniziale periodo oscillatorio.

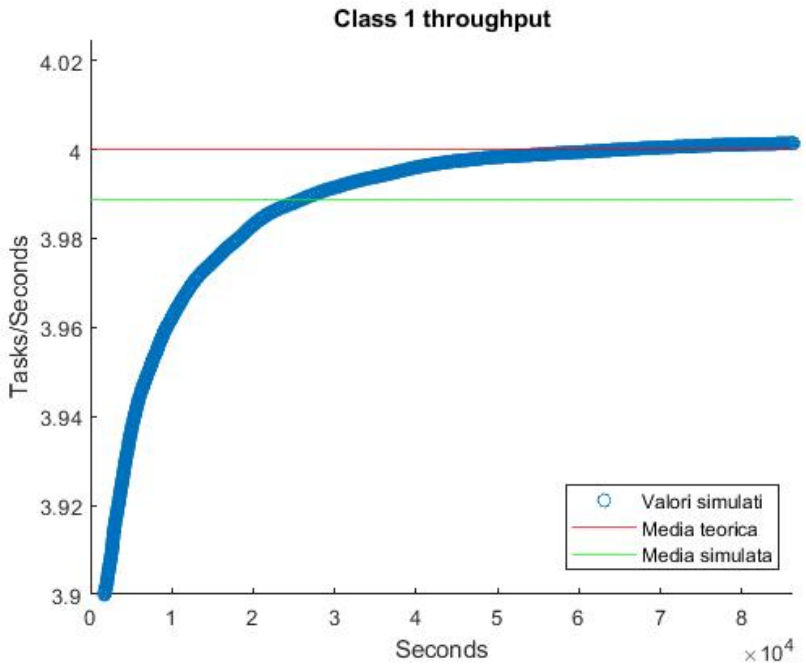


(a) Throughput globale
a: 10.25 task/s, s: 10.231102 (+/-0.01476) task/s

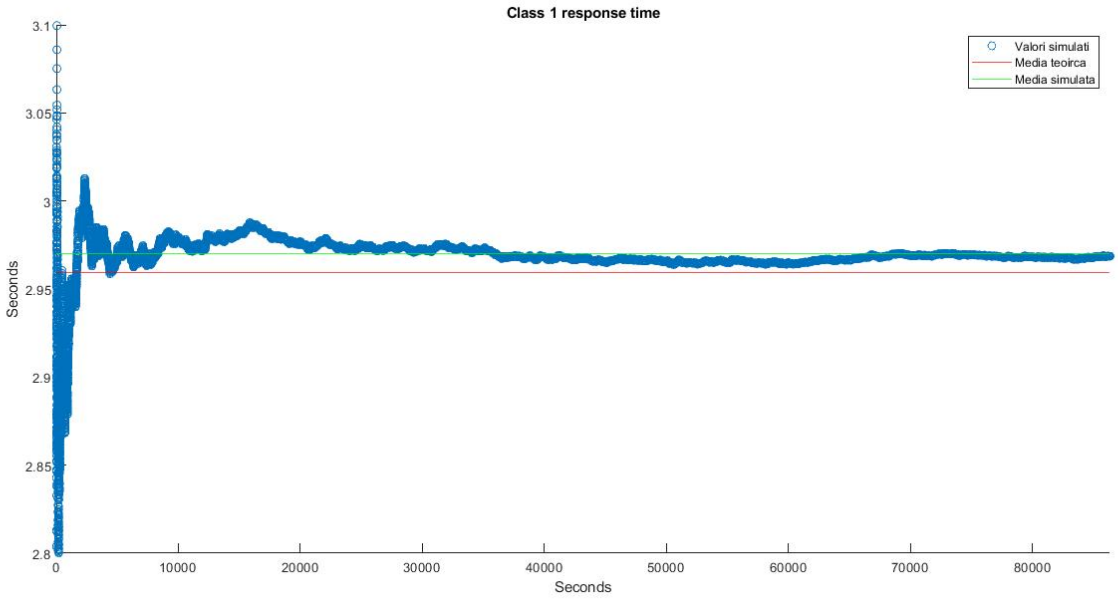


(b) Tempo di risposta globale
a: 3.625938 s, s: 3.622958 (+/-0.00347) s

Figura 8.6: Throughput e tempo di risposta globali

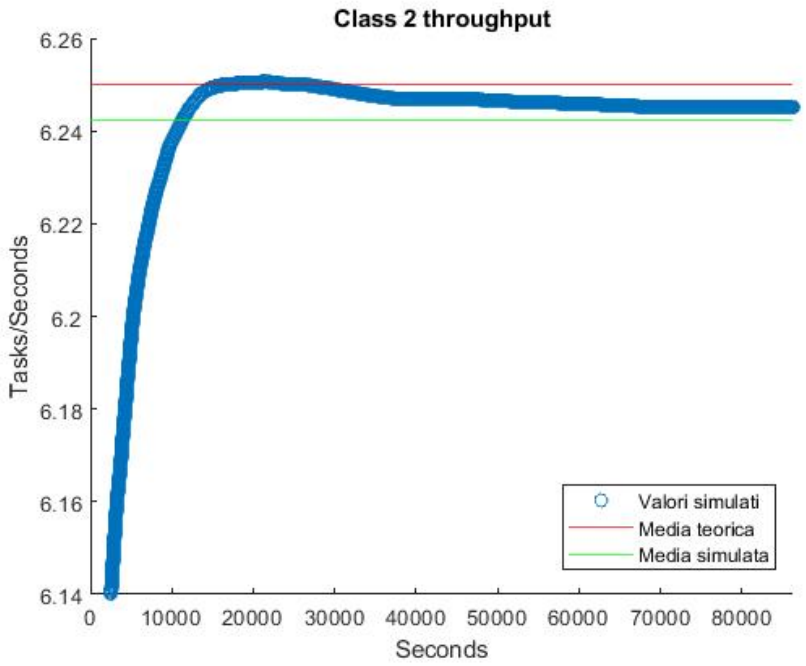


(a) Throughput classe 1
a: 4 task/s, s: 3.988907 (+/-0.00812) task/s

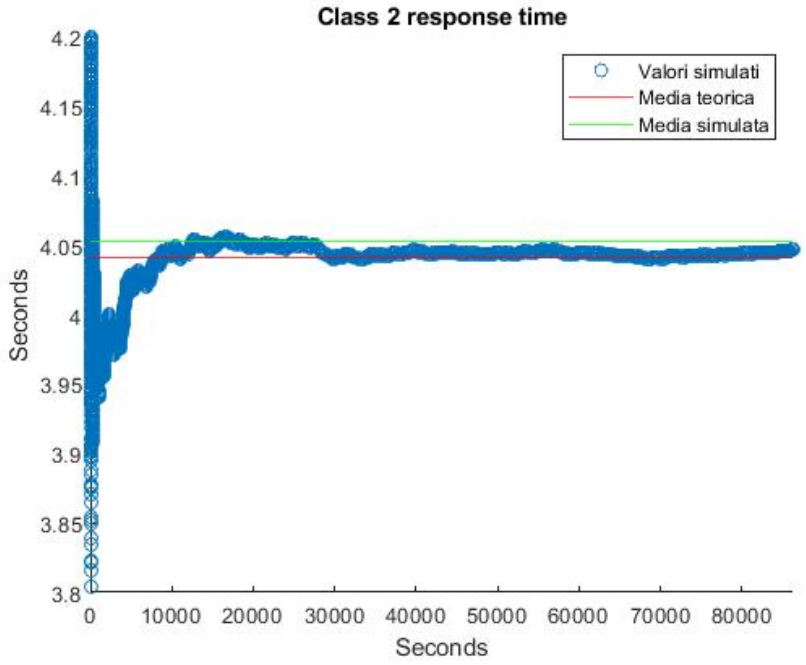


(b) Tempo di risposta classe 1
a: 2.959198 s, s: 2.970106 (+/-0.00184) s

Figura 8.7: Throughput e tempo di risposta globali per classe 1



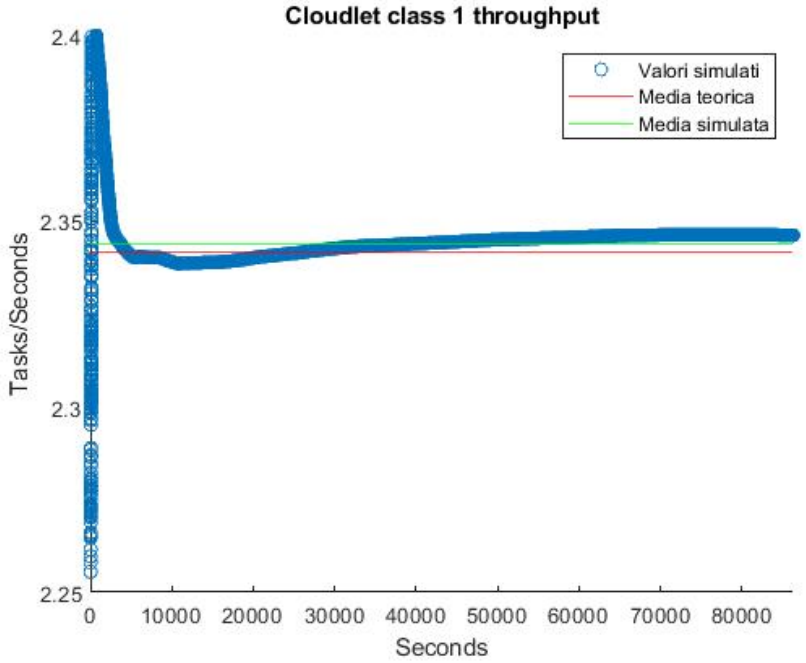
(a) Throughput classe 2
a: 6.25 task/s, s: 6.242194 (+/-) 0.00729 task/s



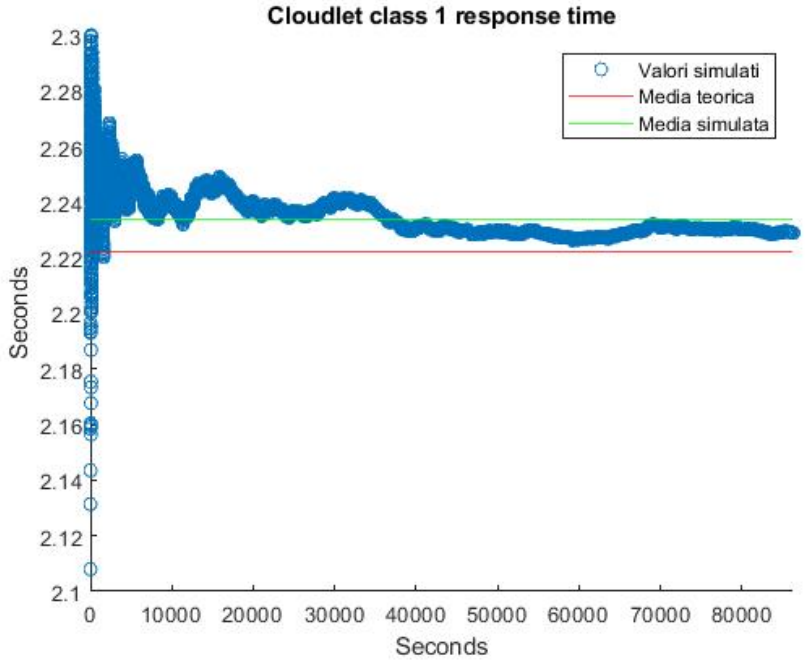
(b) Tempo di risposta classe 2
a: 4.05265 s, s: 4.041308 (+/-0.00582) s

Figura 8.8: Throughput e tempo di risposta globali per classe 2

8.1.2 Throughput e tempo di risposta del cloudlet per classe

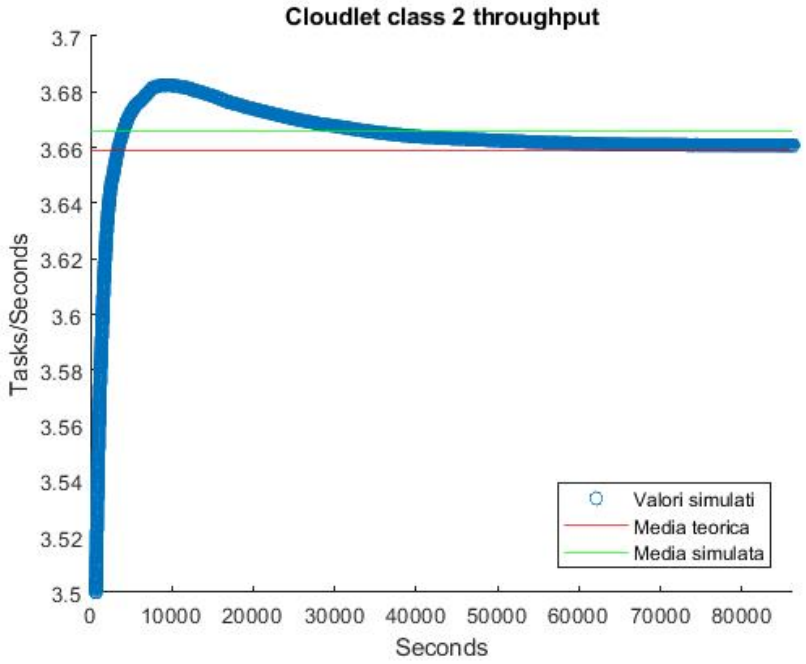


(a) Throughput cloudlet classe 1
a: 2.341804 task/s, s: 2.343732 (+/-0.00110) task/s

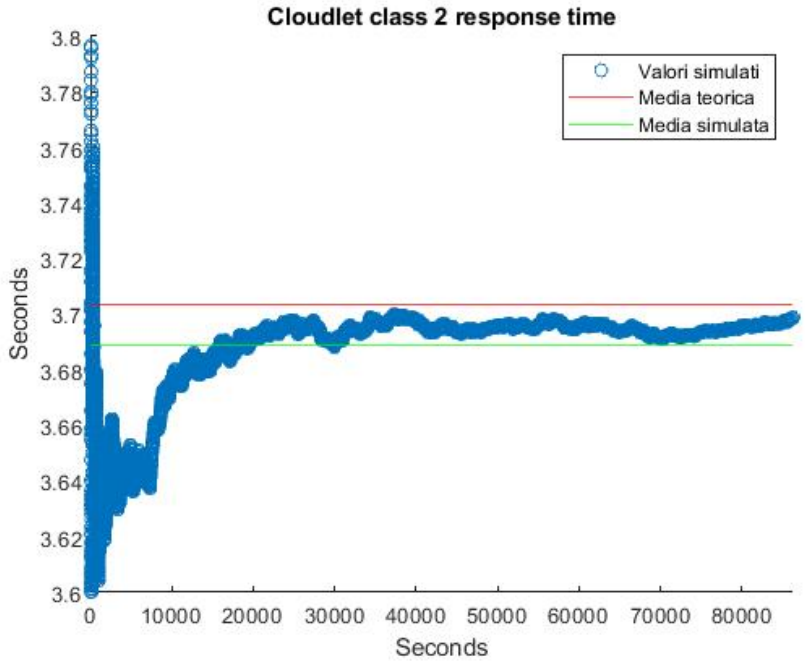


(b) Tempo di risposta cloudlet classe 1
a: $2.\overline{2}s, s : 2.233544(+/-0.00231)s$

Figura 8.9: Throughput e tempo di risposta Classe 1



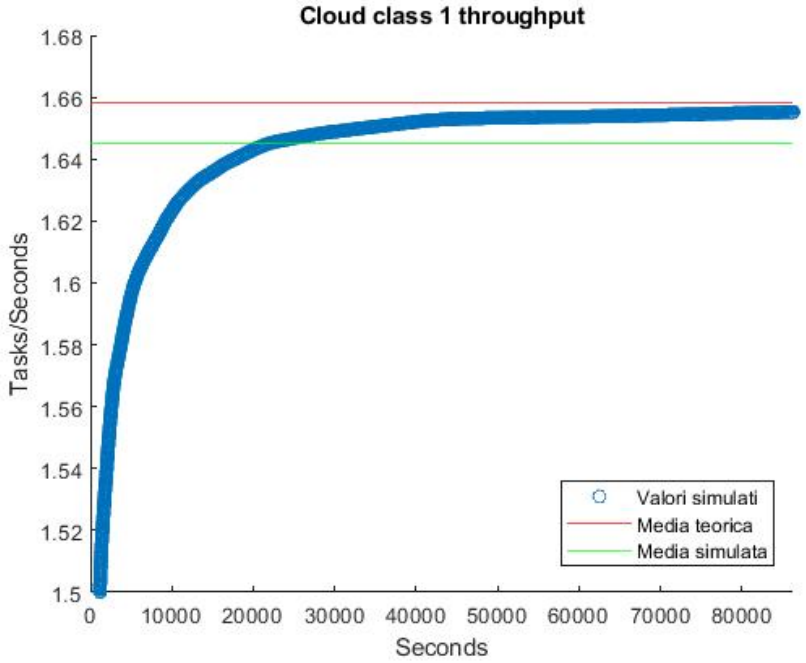
(a) Throughput cloudlet classe 2
a: 3.65906875 task/s, s: 3.666056 (+/-0.00288) task/s



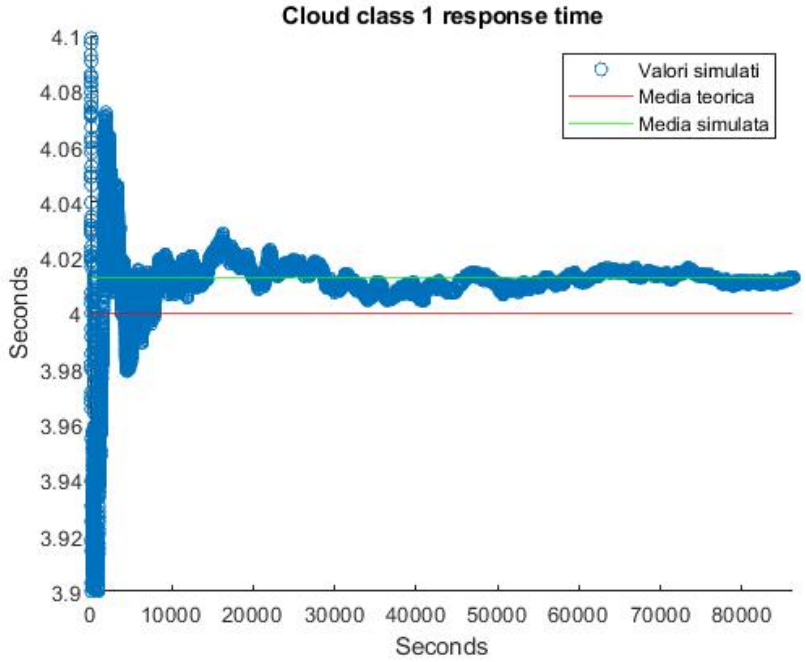
(b) Tempo di risposta cloudlet classe 2
a: $3.\overline{703}s, s : 3.666056(+/-0.00689)s$

Figura 8.10: Throughput e tempo di risposta classe 2

8.1.3 Throughput e tempo di risposta del cloud per classe

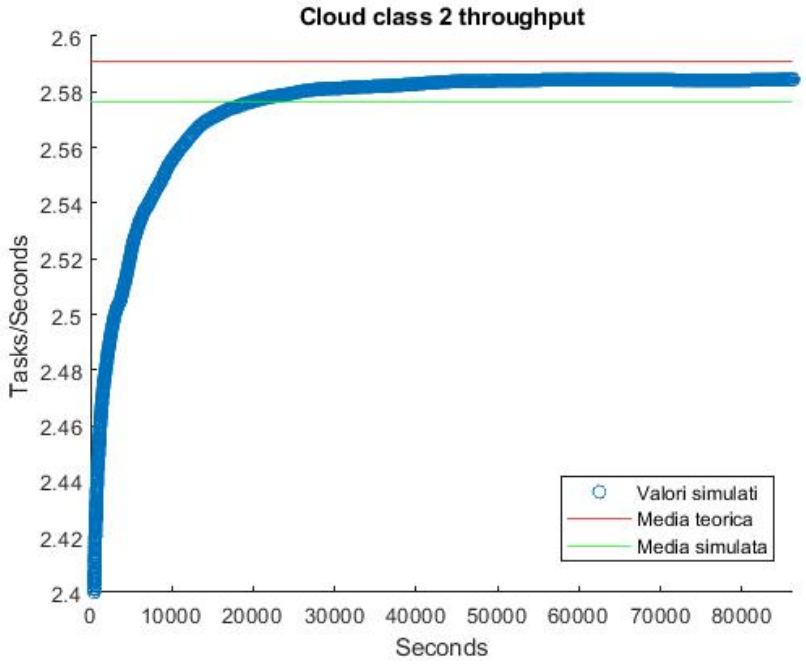


(a) Throughput cloud classe 1
a: 1.658196 task/s, s: 1.645175 (+/-0.00757) task/s

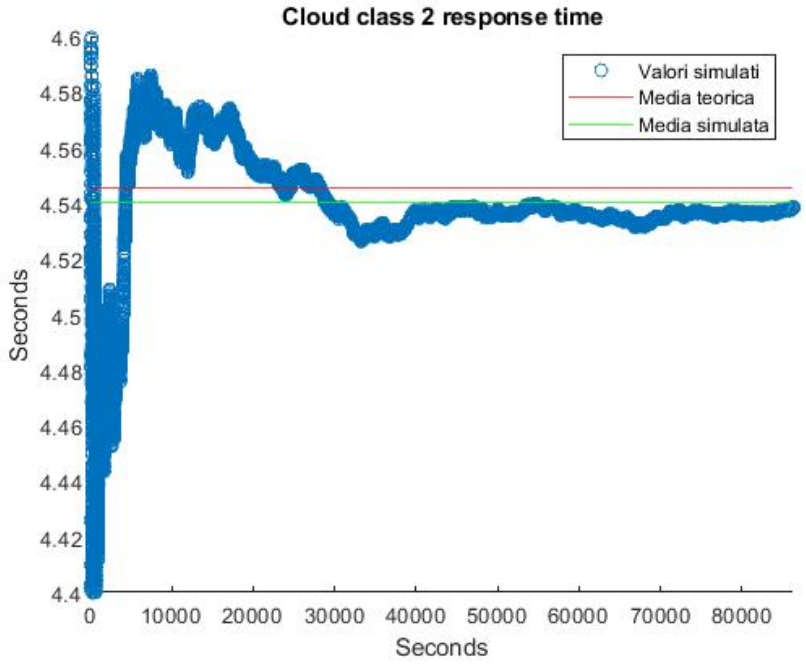


(b) Tempo di risposta cloud classe 1
a: 4 s, s: 4.012712 (+/-0.00148) s

Figura 8.11: Throughput e Tempi di risposta cloud per la classe 1



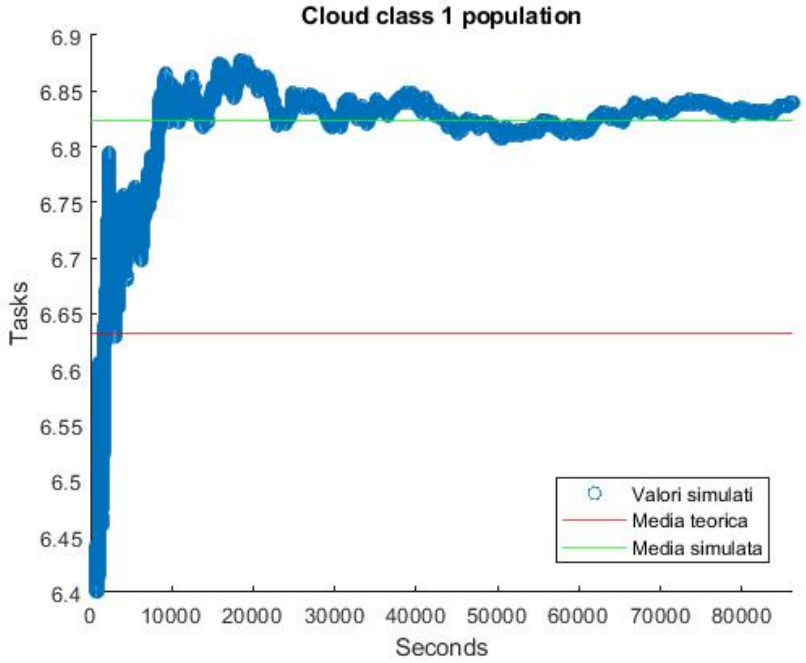
(a) Throughput cloud classe 2
a: 2.59093125 task/s, s: 2.576139 (+/-0.00777) task/s



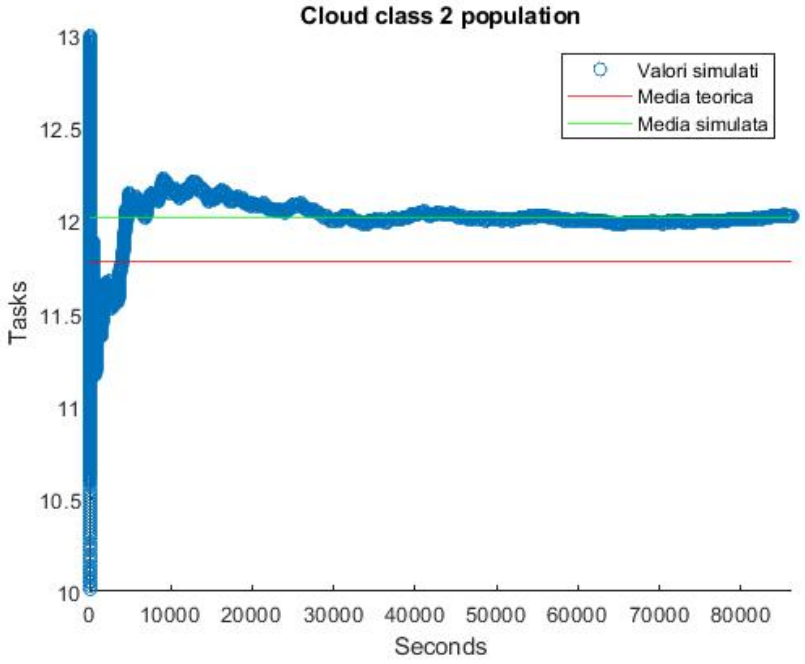
(b) Tempo di risposta cloud classe 2
a: $4.\overline{54}s, s : 4.540414(+/-0.00741)s$

Figura 8.12: Throughput e tempo di risposta cloud Classe 2

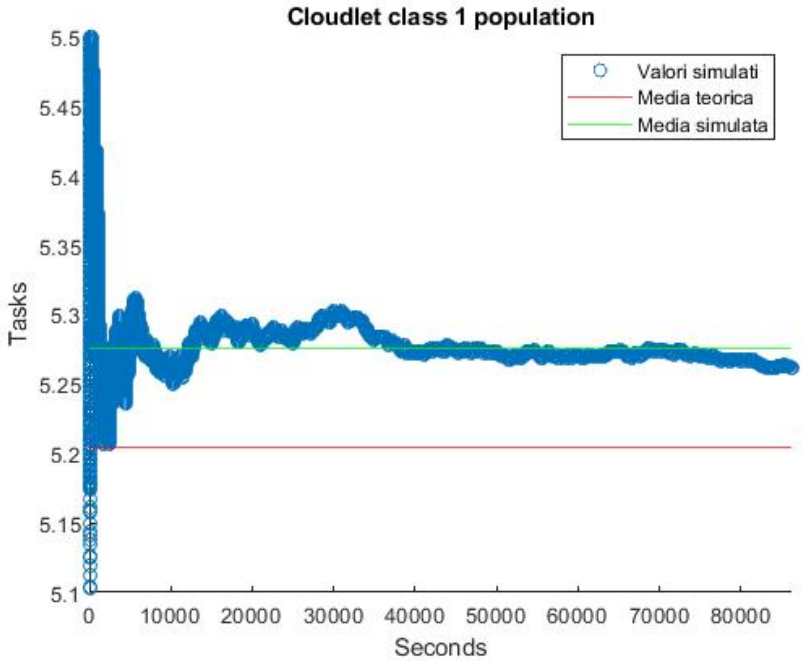
8.1.4 Popolazione media del cloudlet e cloud per classe



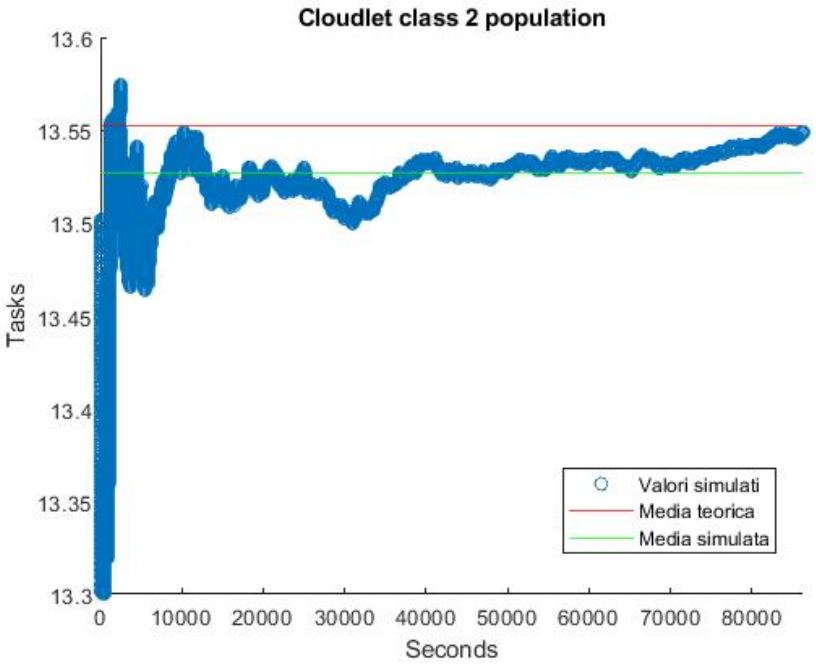
(c) Popolazione media di classe 1 al cloud
a: 0.00864224 task, s: 0.013062 (+/-0.00059) task



(d) Popolazione media di classe 2 al cloud
a: 16.61663 task, s: 17.747327 (+/-0.04850) task



(e) Popolazione media di prima classe al cloudlet
a: 8.884109 task, s: 9.061018 (+/-0.02000) task



(f) Popolazione media di seconda classe al cloudlet
a: 9.608671 task, s: 9.497830 (+/-0.02069) task

Figura 8.9: Tempi di risposta e popolazioni medie per ogni centro

8.1.5 Riepilogo

	Modello analitico	Simulazione
Throughput globale (task/s)	10.25	10.231102 (+/-0.01476)
Throughput globale prima classe (task/s)	4	3.988907 (+/-0.00812)
Throughput globale seconda classe (task/s)	6.25	6.242194 (+/-0.00729)
Tempo di risposta globale (s)	3.625938	3.622958 (+/-0.00347)
Tempo di risposta prima classe (s)	2.2959198	2.2970106 (+/-0.00184)
Tempo di risposta seconda classe (s)	4.05265	4.041308 (+/-0.00582)
Throughput globale al cloudlet (task/s)	6.00087275	6.009788 (+/-0.00184)
Throughput prima classe al cloudlet (task/s)	2.341804	2.343732 (+/-0.00110)
Throughput seconda classe al cloudlet (task/s)	3.65906875	3.666056 (+/-0.00288)
Throughput prima classe al cloud (task/s)	1.658196	1.645175 (+/-0.00757)
Throughput seconda classe al cloud (task/s)	2.59093125	2.576139 (+/-0.00777)
Tempo di risposta prima classe al cloud (s)	4	4.012712 (+/-0.00148)
Tempo di risposta seconda classe al cloud (s)	$4.\overline{54}$	4.540414 (+/-0.00741)
Tempo di risposta prima classe al cloudlet (s)	$2.\overline{2}$	2.233544 (+/-0.00231)
Tempo di risposta seconda classe al cloudlet (s)	$3.\overline{307}$	3.666056 (+/-0.00689)
Popolazione media prima classe al cloud (task)	6.632784	6.823794 (+/-0.01430)
Popolazione media seconda classe al cloud (task)	11.776958	12.016666 (+/-0.04330)
Popolazione media prima classe al cloudlet (task)	5.204008	5.275428 (+/-0.00411)
Popolazione media seconda classe al cloudlet (task)	13.552107	13.526597 (+/-0.00556)

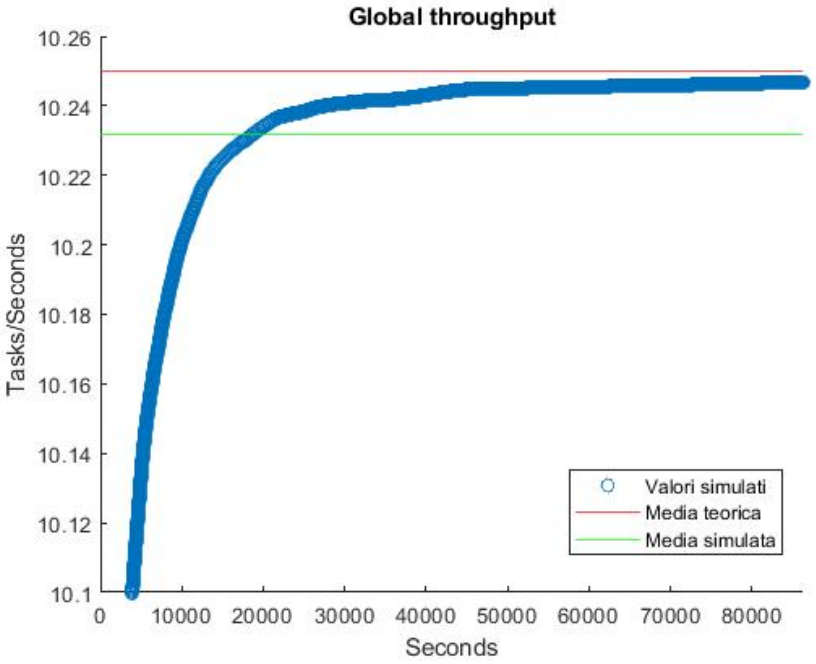
8.2 Statistiche per l'algoritmo 2

Parametri di simulazione

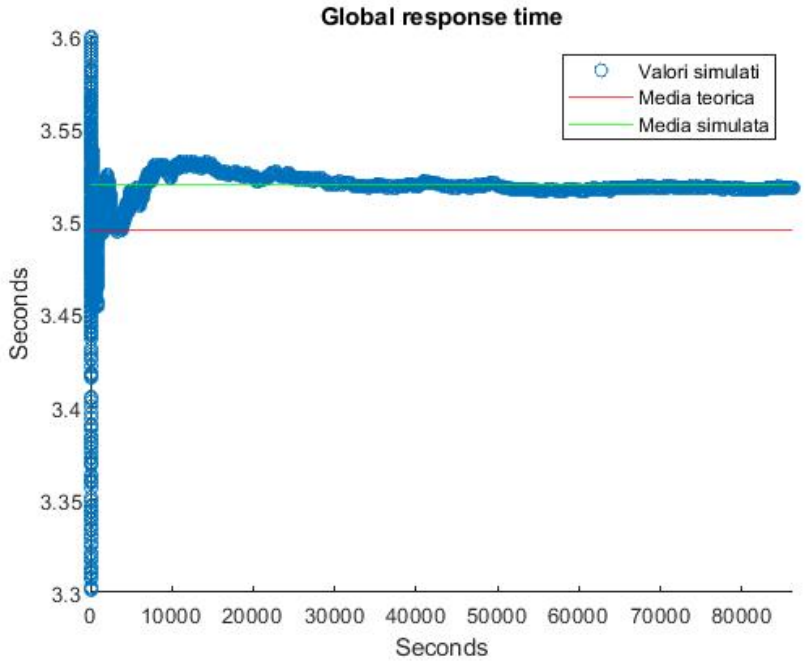
- Tempo totale: 86400 secondi
- Seed iniziale: 123456
- Dimensione del batch: 3600 secondi
- Scheduling: Random
- Soglia: 20

8.2.1 Tempo di risposta e throughput globale e per classe

Anche per quanto riguarda l'algoritmo 2 possiamo vedere come i throughput globale e per classi convergono alle medie teorica. Analogamente i tempi di risposta globale e per classi convergono ai valori analizzati nel modello analitico, che come ci si aspettava ha subito una riduzione rispetto all'algoritmo precedente.
Da questi risultati si può notare che il tempo di risposta per task di classe 1 sia minore rispetto agli altri, poiché in tale classe non vi sono job prelazionabili. Questa differenza era già presente nell'algoritmo 1 ma, nell'algoritmo 2, è stata amplificata proprio per la possibilità di prelazionare i task.

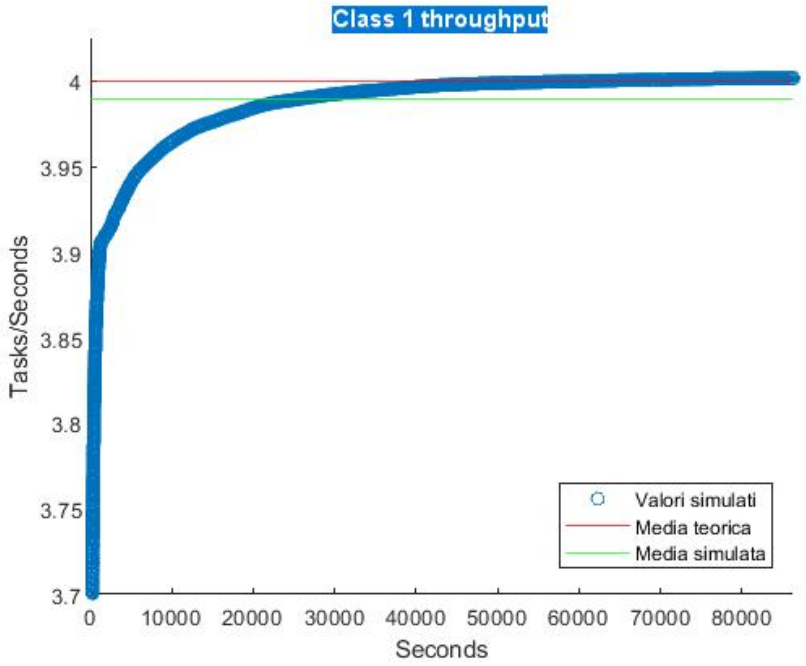


(g) Throughput globale
a: 10.25 task/s, s: 10.231724 (+/-0.01448) task/s

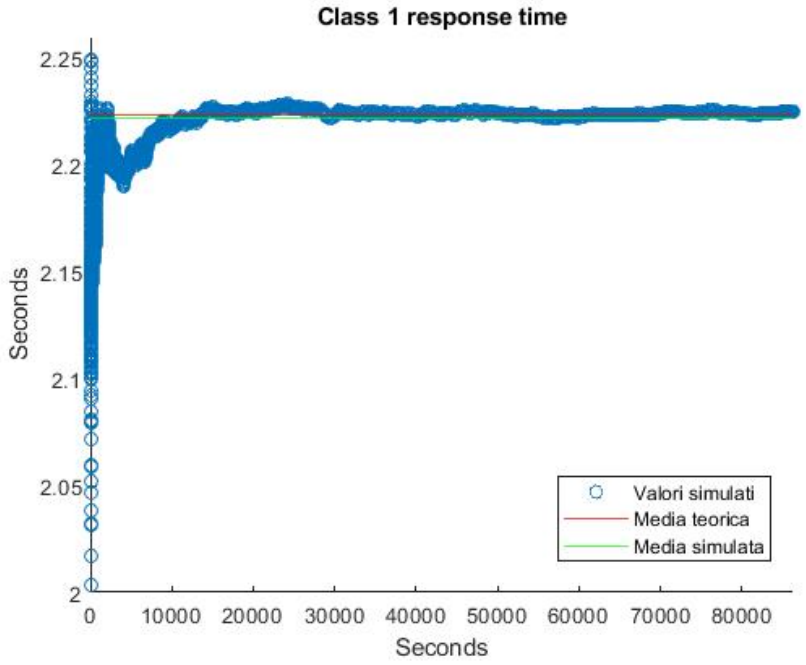


(h) Tempo di risposta globale
a: 3.495674 s, s: 3.458544 (+/-0.00237) s

Figura 8.10: Throughput e tempo di risposta globali

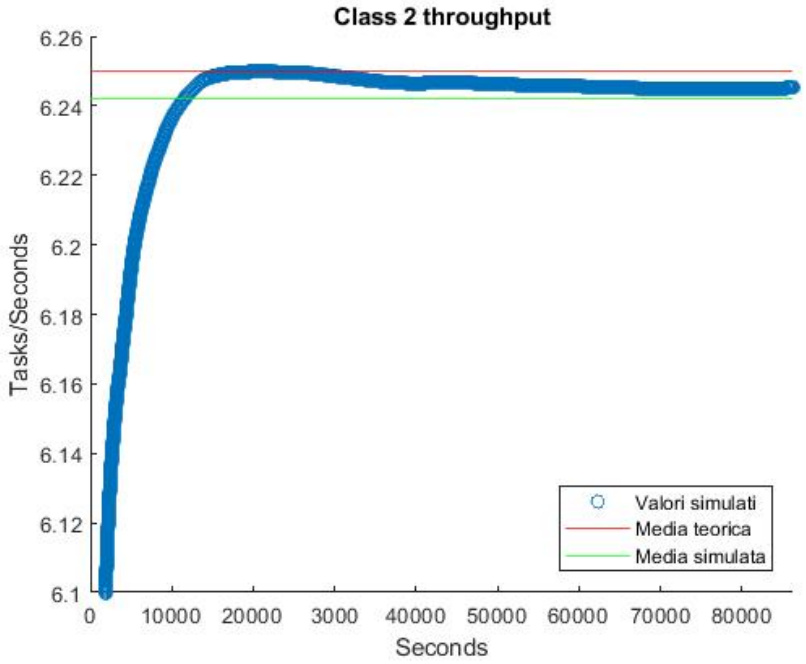


(a) Throughput globale della prima classe
a: 4 task/s, s: 3.989873 (+/-0.00768) task/s

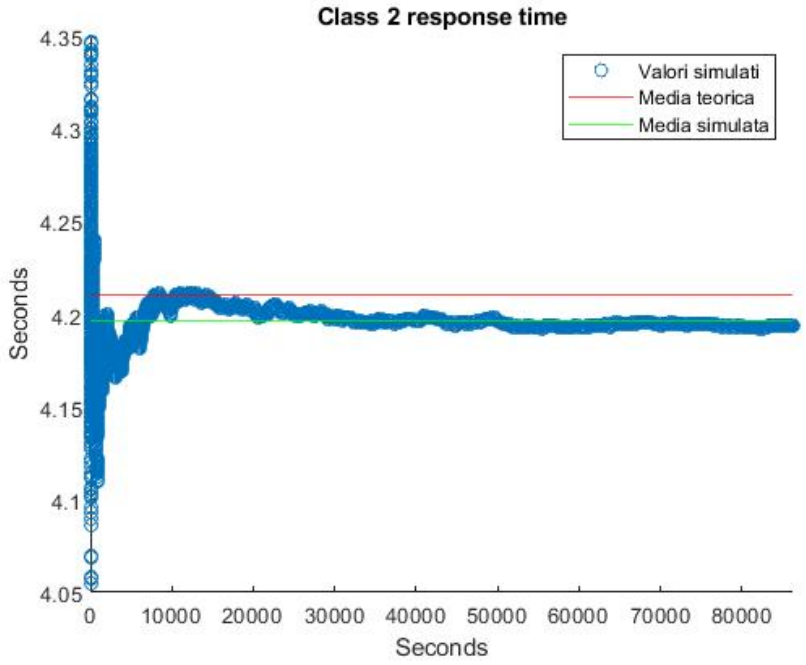


(b) Tempo di risposta della prima classe
a: $2.\overline{2}s, s : 2.222094(+/-0.00274)s$

Figura 8.11: Throughput e tempo di risposta per task di tipo 1



(a) Throughput globale della seconda classe
a: 6.25 task/s, s: 6.241851 (+/-0.00744) task/s

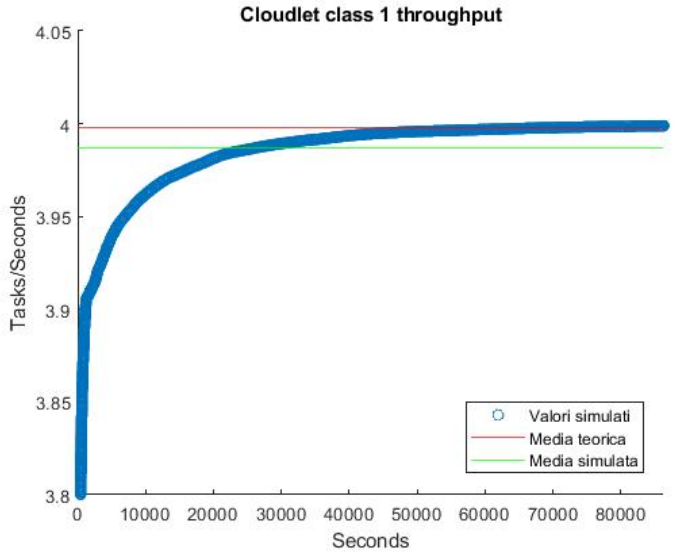


(b) Tempo di risposta globale della seconda classe
a:4.210071, s: 4.103553 (+/-0.00254) s

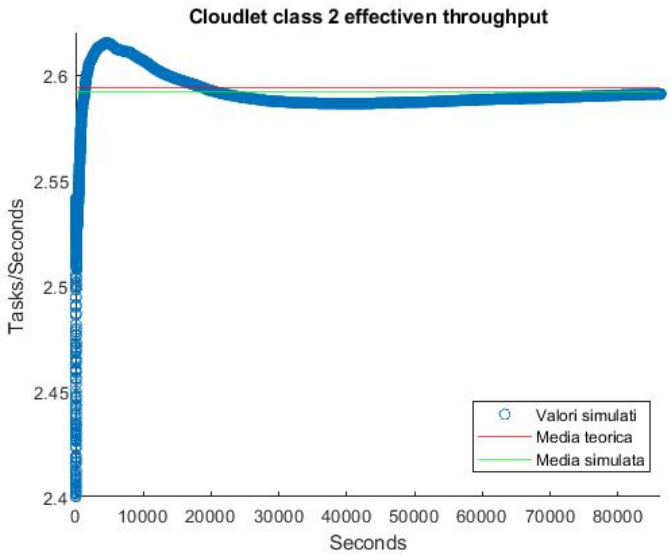
Figura 8.12: Throughput e tempo di risposta per task di tipo 2

8.2.2 Throughput effettivo e tempo di risposta del cloudlet per classe

Il throughput al cloudlet è stato calcolato non contando i task prelazionati, in quanto completati dal cloud. Come si può notare dai grafici tutte le metriche considerate convergono ad un valore, che in questo caso è anche il valore teorico calcolato analiticamente.
Si può notare come nel grafico (b) il throughput effettivo di classe 2 nel cloudlet tende al valore analitico dopo una fase iniziale in cui vi è un numero di task/secondo maggiore. Questo è dovuto al fatto che tali task vengono prelazionati.



(a) Throughput cloudlet classe 1
a: 3.997849 task/s, s: 3.987055 (+/-0.00750) task/s

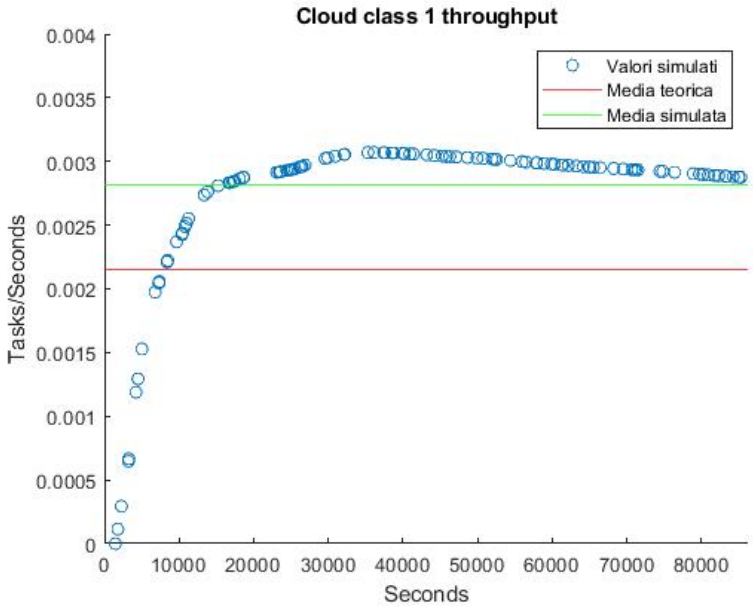


(b) Throughput cloudlet classe 2
a: 2.594341 task/s, s: 2.591935 (+/-0.00322) task/s

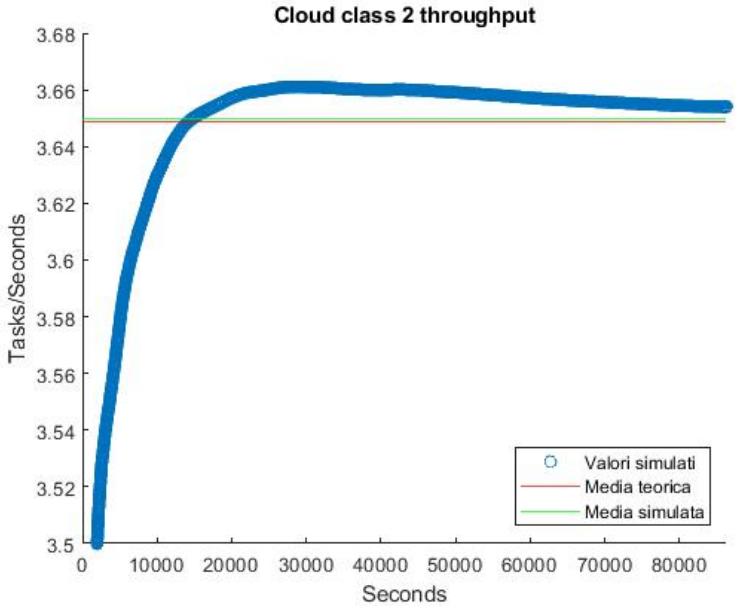
Figura 8.13: Throughput cloudlet per classe 1 e 2

8.2.3 Throughput del cloud per classe

Il throughput di classe 1 nel cloud tende al valore analitico, tuttavia come si può notare dal grafico (a) il numero di misurazioni è molto inferiore rispetto al solito, questo proprio perchè i task di classe 1 sono eseguiti principalmente nel cloudlet a causa della prelazione.



(a) Throughput cloud classe 1
a: 0.00215106 task/s, s: 0.002818 (+/-0.00020) task/s



(b) Tthroughput cloud classe 2
a: 3.648699075 task/s, s: 3.649916 (+/-0.01014) task/s

Figura 8.14: Throughput cloud per la classe 1 e 2

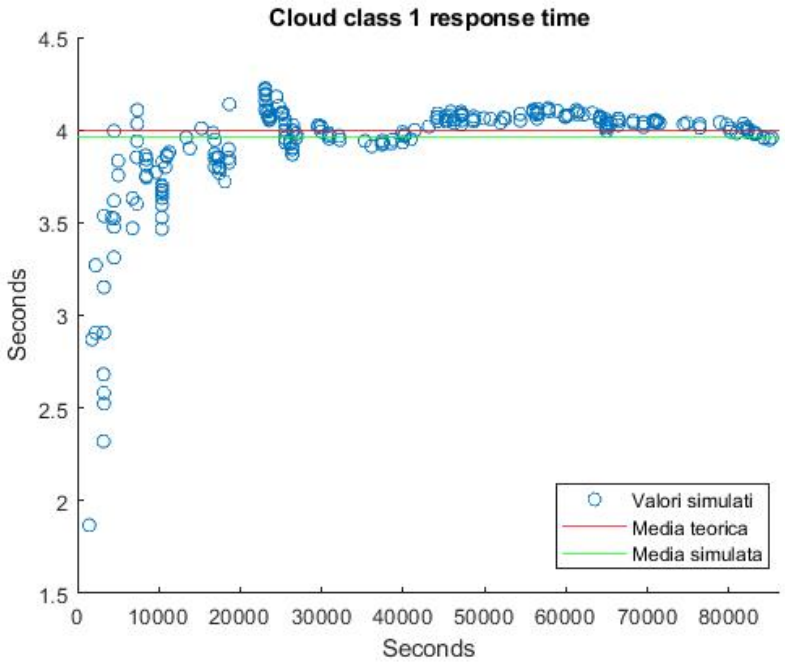
8.2.4 Tempo di risposta e popolazione media per ogni centro

Tempi di risposta

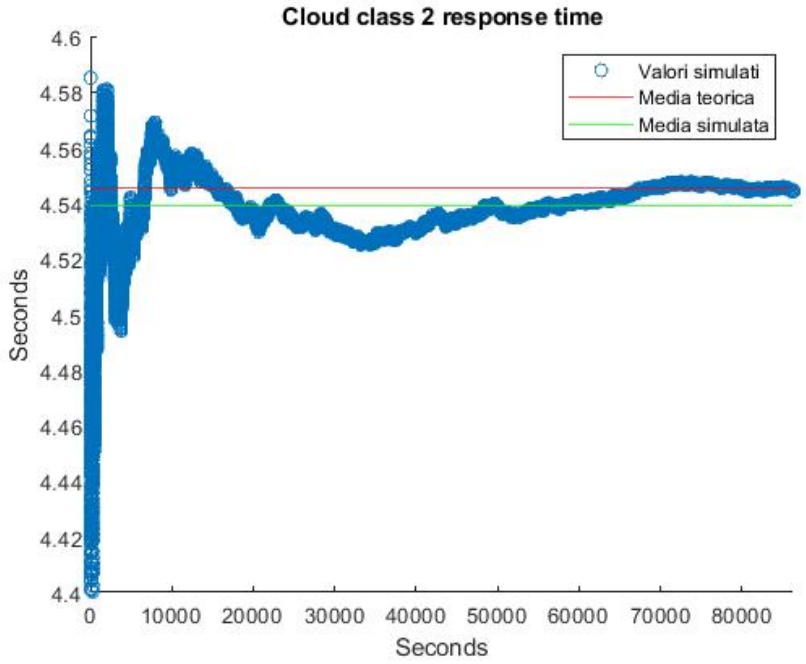
Dai grafici dei tempi di risposta è possibile vedere come questi convergano alla media teorica.

Popolazioni

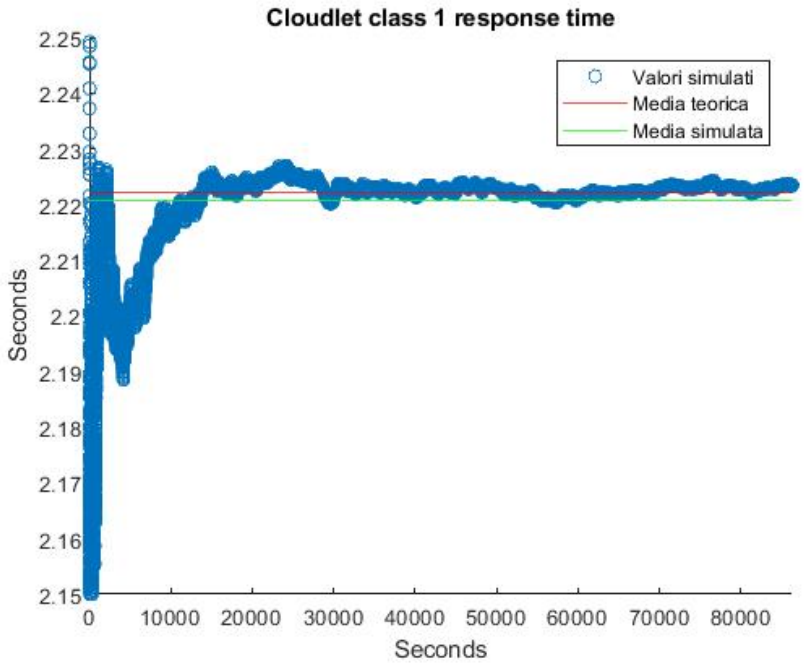
Per le popolazioni si può notare come queste convergano ad un valore vicino alla stima teorica a meno di un errore. Abbiamo notato questa differenza in tutti i casi, ma non siamo arrivati ad una conclusione certa sul motivo di questo fenomeno.



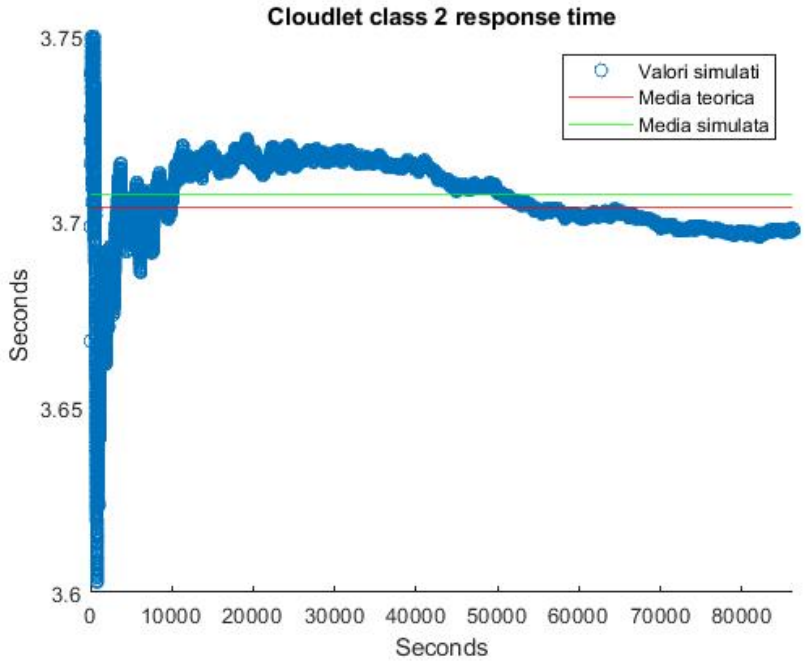
(a) Tempo di risposta al cloud per task di tipo 1
a: 4 s, s: 4.130490 (+/-0.171825) s



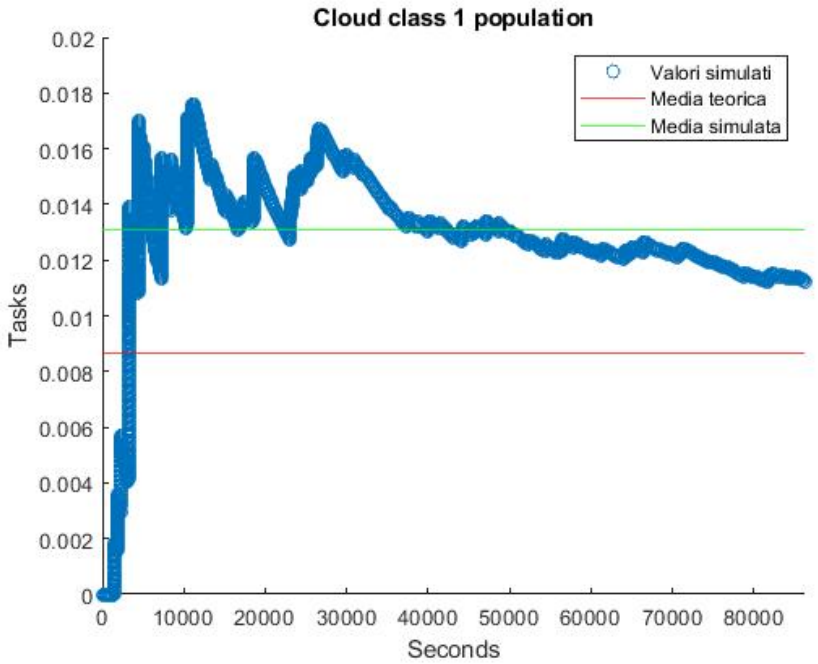
(b) Tempo di risposta al cloud per task di tipo 2
a:4.5454 s, s: 4.538889 (+/-0.00422) s



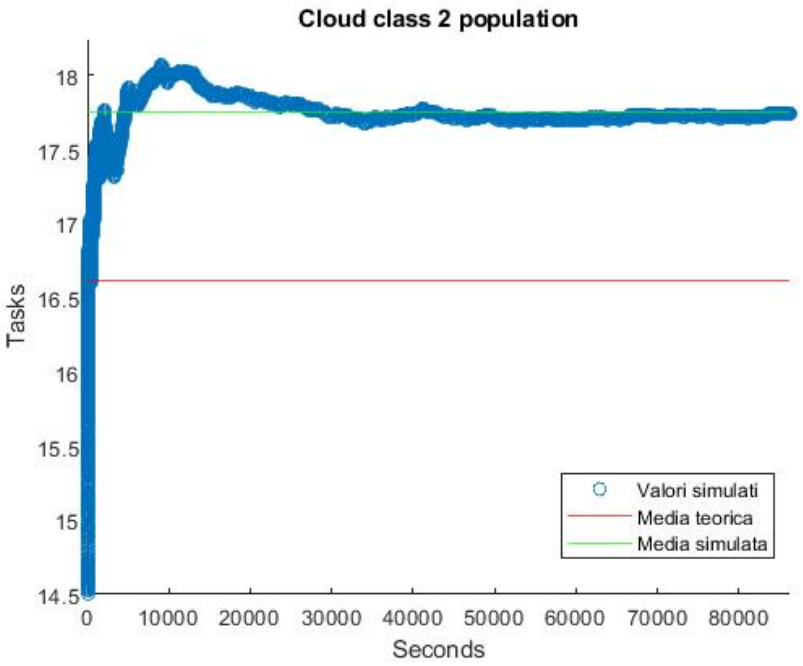
(c) Tempo di risposta al cloudlet per task di tipo 1
a:$2.\overline{2}s, s : 2.218467(+/-0.007387)s$



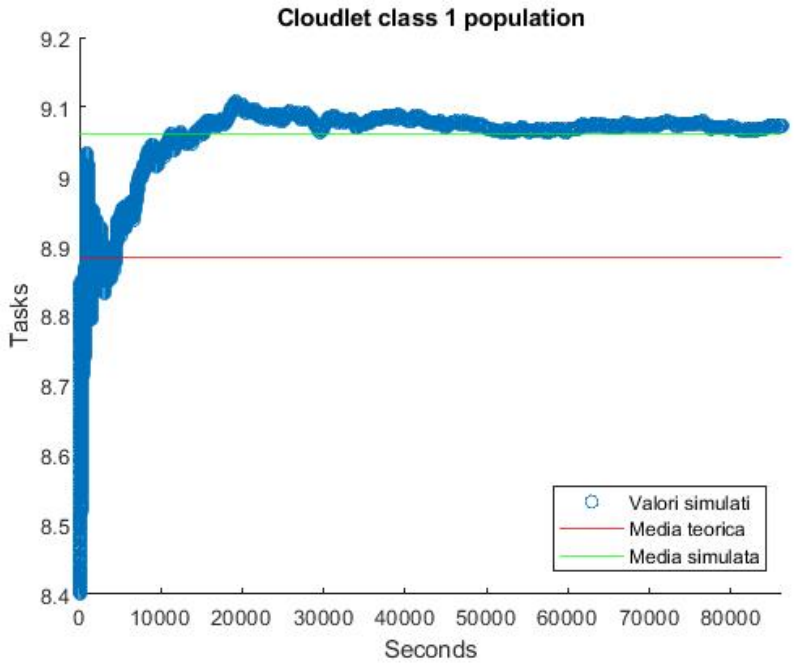
(d) Tempo di risposta al cloudlet per task di tipo 2
a:3.7037 s, s: 2.426817 (+/-0.016152) s



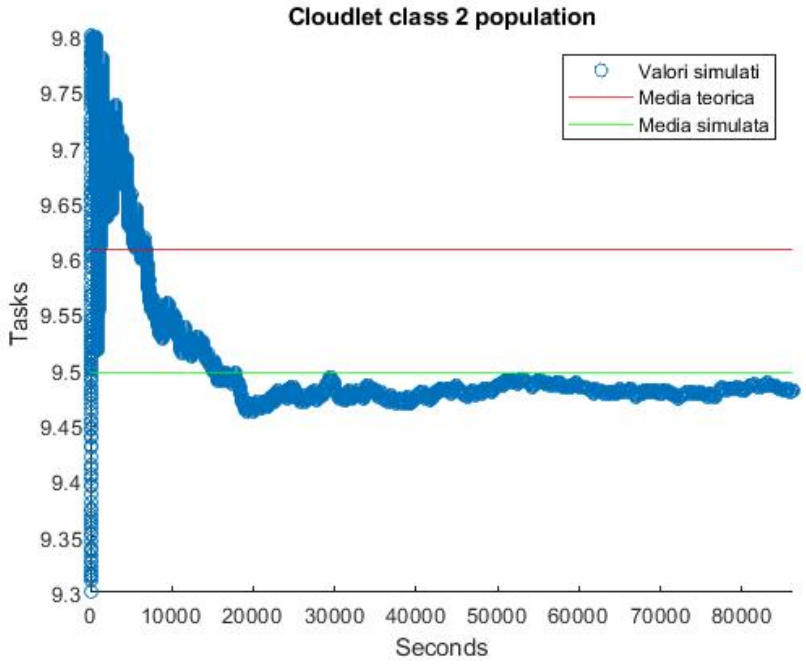
(e) Popolazione media di classe 1 al cloud
a: 0.00864224 task ,s: 0.013062 (+/-0.00059) task



(f) Popolazione media di classe 2 al cloud
a: 16.61663 task, s: 17.747327 (+/-0.04850) task



(g) Popolazione media di prima classe al cloudlet
a: 8.884109 task, s: 9.061018 (+/-0.02000) task



(h) Popolazione media di seconda classe al cloudlet
a: 9.608671 task, s: 9.497830 (+/-0.02069) task

Figura 8.8: Tempi di risposta e popolazioni medie per ogni centro

8.2.5 Riepilogo

S = 20	Modello analitico	Simulazione
Throughput globale (task/s)	10.25	10.231724 (+/-0.01448)
Throughput globale prima classe (task/s)	4	3.989873 (+/-0.00768)
Throughput globale seconda classe (task/s)	6.25	6.241851 (+/-0.00744)
Tempo di risposta globale (s)	3.495674	3.458544 (+/-0.00237)
Tempo di risposta prima classe (s)	$2.\overline{2}$	2.222094 (+/-0.00274)
Tempo di risposta seconda classe (s)	4.210071	4.103553 (+/-0.00254)
Throughput prima classe al cloud (task/s)	0.00215106	0.002818 (+/-0.00020)
Throughput seconda classe al cloud (task/s)	3.648699075	3.649916 (+/-0.01014)
Throughput effettivo prima classe al cloudlet (task/s)	3.997849	3.987055 (+/-0.00750)
Throughput effettivo seconda classe al cloudlet (task/s)	2.594341	2.591935 (+/-0.00322)
Tempo di risposta prima classe al cloud (s)	4	4.130490 (+/-0.0171825)
Tempo di risposta seconda classe al cloud (s)	$4.\overline{54}$	4.538889 (+/-0.00422)
Tempo di risposta prima classe al cloudlet (s)	$2.\overline{2}$	2.218467 (+/-0.007387)
Tempo di risposta seconda classe al cloudlet (s)	3.7037	2.426817 (+/-0.016152)
Popolazione media prima classe al cloud (task)	0.00864224	0.013027 (+/-0.00059)
Popolazione media seconda classe al cloud (task)	16.61663	17.029314 (+/-0.04705)
Popolazione media prima classe al cloudlet (task)	8.884109	9.070531 (+/-0.01993)
Popolazione media seconda classe al cloudlet (task)	9.608671	9.493828 (+/-0.02075)

8.2.6 Miglioramento algoritmo 2 nei task prelazionati

Nell'algoritmo 2 i task che vengono prelazionati e spostati nel cloud oltre ad avere un tempo di setup eseguono il task ripartendo dallo stato iniziale. Questo comporta un peggioramento del tempo di risposta globale. Una possibile miglioria è rappresentata dalla possibilità di far continuare l'esecuzione dal momento in cui era stata interrotta.
Dal punto di vista analitico non è stato possibile modellare questa variante dell'algoritmo, tuttavia dal punto di vista della simulazione, conoscendo i tempi di servizio, abbiamo notato che il tempo di risposta globale si riduceva notevolmente. Per permettere lo spostamento senza alcuna perdita di tempo si è utilizzata una euristica che permettesse di approssimare il tempo effettivo rimanente da eseguire sul cloud. Questo è stato fatto tramite una proporzione che tenesse conto del diverso tasso di servizio tra cloud e cloudlet(μ_{cloud} e μ_{clet}).
In particolare la formula utilizzata è:

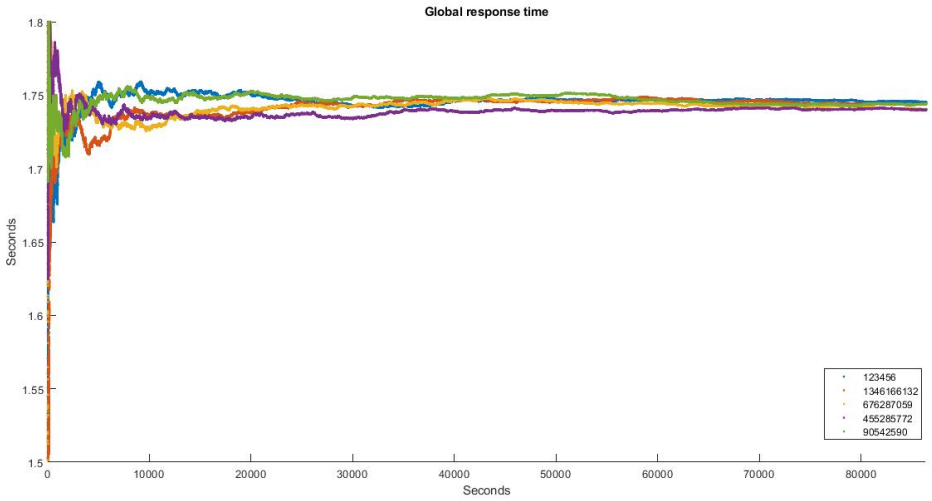
$$\frac{\mu_{cloud}}{\mu_{clet}} * RemainingTime + Setup \tag{8.11}$$

Si può notare il miglioramento nei tempi di risposta, a parità di soglia S pari a 20, nella seguente tabella:

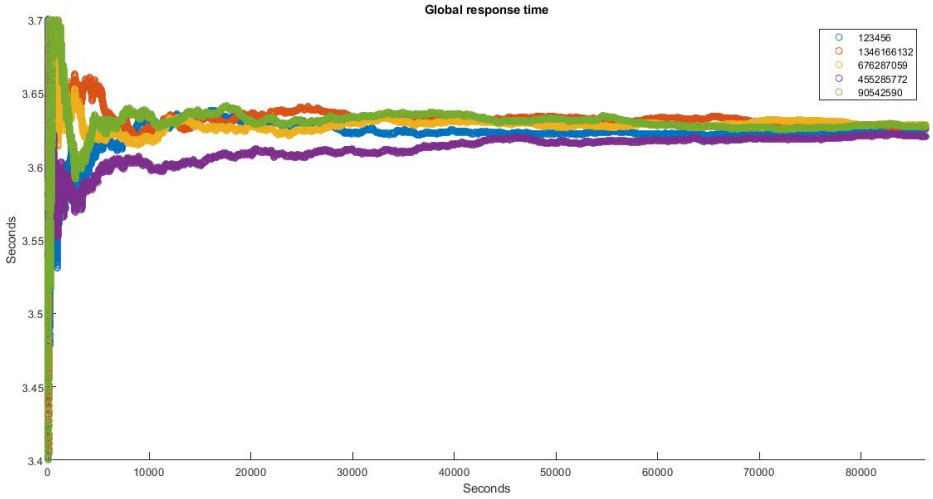
	Algoritmo 2	Variante algoritmo 2
Tempo di risposta globale (s)	3.458544 (+/-0.00237)	3.330771 (+/-0.00150)
Tempo di risposta prima classe (s)	2.222094 (+/-0.00274)	2.218547 (+/-0.00207)
Tempo di risposta seconda classe (s)	4.103553 (+/-0.00254)	3.911301 (+/-0.00242)
Tempo di risposta prima classe al cloud (s)	4.130490 (+/-0.00020)	4.213663 (+/-0.11359)
Tempo di risposta seconda classe al cloud (s)	4.538889 (+/-0.01014)	4.146376 (+/-0.00244)
Tempo di risposta prima classe al cloudlet (s)	2.218467 (+/-0.00750)	2.217407 (+/-0.00202)
Tempo di risposta seconda classe al cloudlet (s)	2.426817 (+/-0.016152)	3.698034 (+/-0.00290)

8.3 Confronto caso cloudlet iperesponenziale e esponenziale

Nella traccia veniva richiesto di effettuare il confronto del modello analitico con quello simulato considernando il cloudlet con un tempo di servizio esponenziale anziché iperseponenziale per semplificare l'analisi. Nel caso del cloudlet iperesponenziale si ottengo tempi di risposta globali nettamente migliori rispetto al caso esponenziale. Questo è proprio dovuto al fatto che il servizio è più efficiente.
Di seguito vediamo il confronto tra i due modelli con le differenti distribuzioni, sia per l'algoritmo 1 che per l'algoritmo 2.
Ogni risultato è stato calcolato con metodo di replicazione dei run di simulazione con cinque seed diversi: 123456 (blu), 1346166132 (rosso),676287059 (giallo), 455285772 (viola) e 90542590 (verde).

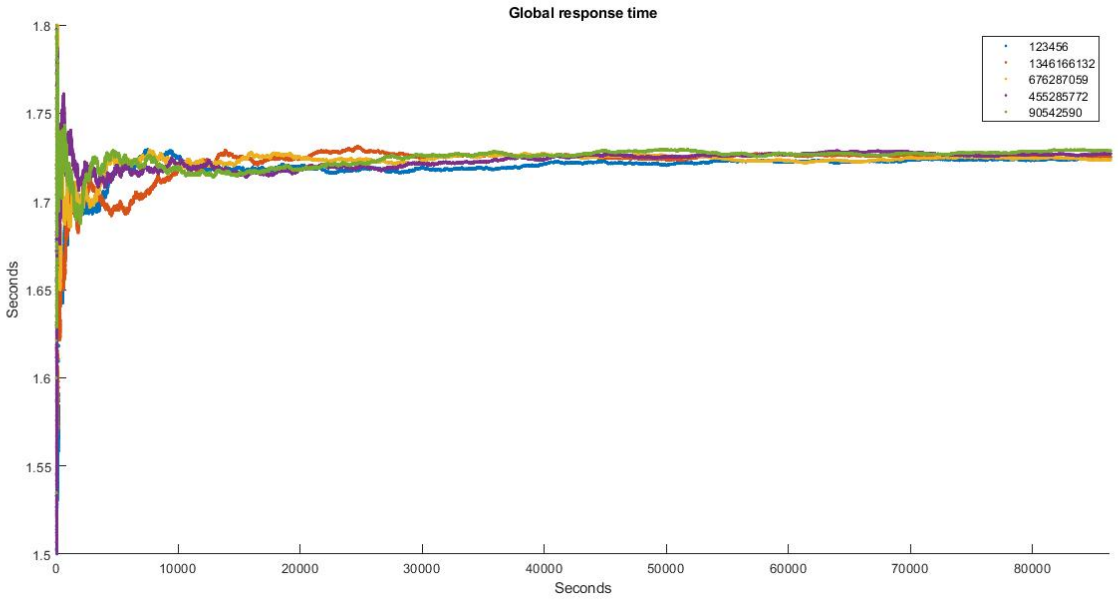


(i) Tempo di risposta globale algoritmo 1 caso cloudlet iperesponenziale

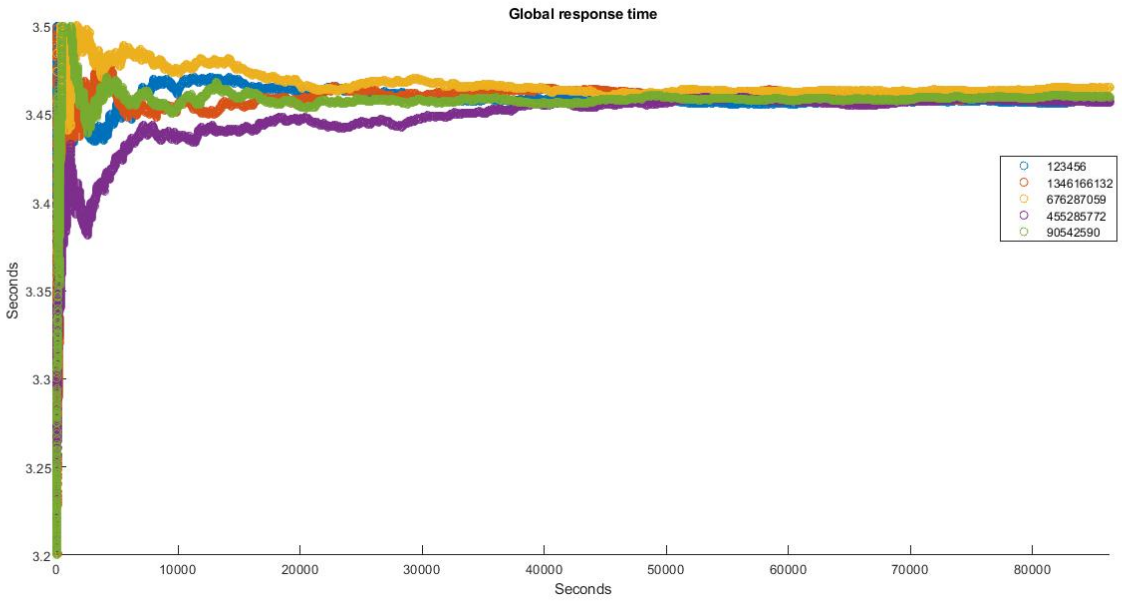


(j) Tempo di risposta globale algoritmo 1 caso cloudlet esponenziale

Figura 8.9: Statistiche a regime algoritmo 1



(a) Tempo di risposta globale algoritmo 2 caso cloudlet iperesponenziale



(b) Tempo di risposta globale algoritmo 2 caso cloudlet esponenziale

Figura 8.10: Statistiche a regime algoritmo 2

Come possiamo dedurre dalle figure con entrambe le distribuzioni il sistema risulta essere stazionario e il tempo di risposta nel caso iperesponenziale tende a stabilizzarsi a valori di circa 1.75 s per l'algoritmo 1 e 1.70 s per l'algoritmo 2.

Studio delle distribuzioni del throughput

9.0.1 Interarrivi del cloudlet - Distribuzioni

Essendo presente un controllore a gestire gli arrivi abbiamo deciso di analizzare la distribuzione di questi ultimi, sia per il cloud che per il cloudlet, al fine di verificare di che distribuzione si trattasse. Le analisi degli interarrivi sono state effettuate direttamente sull'algoritmo 2 in quanto l'algoritmo 1 risutava essere abbastanza simile, se non ancor più semplificato.
Abbiamo simulato il sistema per 86400 secondi con il valore di soglia S pari a 20, con lo scheduling RANDOM per quanto riguarda la scelta dei task da prelazionare e dimensione del batch di 3600 secondi (seed fissato a 123456).
Il risultato ottenuto dall'istogramma creato a partire dalle frequenze dei valori di interarrivo dei task al cloudlet è quello di una, con buona approssimazione, variabile aleatoria esponenziale. Questo risultato della simulazione conferma le ipotesi dello studio analitico che voleva gli arrivi casuali al cloudlet come markoviani. Infatti, essendo questi ultimi costituiti da due flussi esponenziali indipendenti di parametro $\lambda_{1,clet} = 3,997849 task/s$ e $\lambda_{2,clet} = 2,594341 task/s$ la distribuzione aspettata sarebbe data dal minimo dei due.
La nuova variabile aleatoria ottenuta si distribuisce ancora come un'esponenziale il cui parametro è ottenuto come somma dei parametri delle due distribuzioni di partenza:

$$P(min(T_1, T_2) > t) = P(T_1 > t, T_2 > t) = P(T_1 > t) \cdot P(T_2 > t) = \\ = e^{-\lambda_1 t} \cdot e^{-\lambda_2 t} = e^{-(\lambda_1 + \lambda_2)t} \quad (9.12)$$

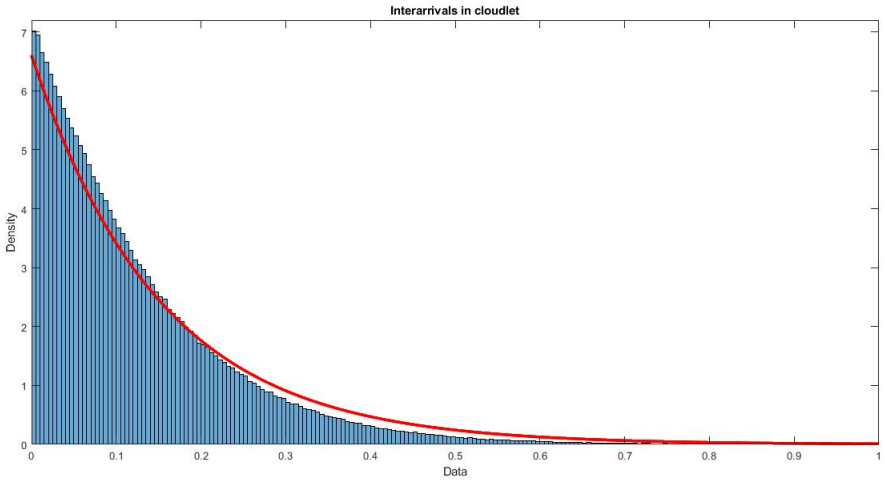


Figura 9.11: interarrivi al cloudlet

9.0.2 Interarrivi del cloud - Distribuzioni

Per quanto riguarda il tasso di arrivo dei task al cloud, a seguito di un'analisi della loro distribuzione, possiamo vedere dal grafico che non risulta seguire un andamento esponenziale. Rispetto al precedente curve-fitting ricavato per il sistema cloudlet, infatti è possibile vedere come questo non si adatti alla funzione scelta. Pertanto nel modello analitico ci limitiamo a definire il sistema cloud come un infinite server $G/M/\infty$.

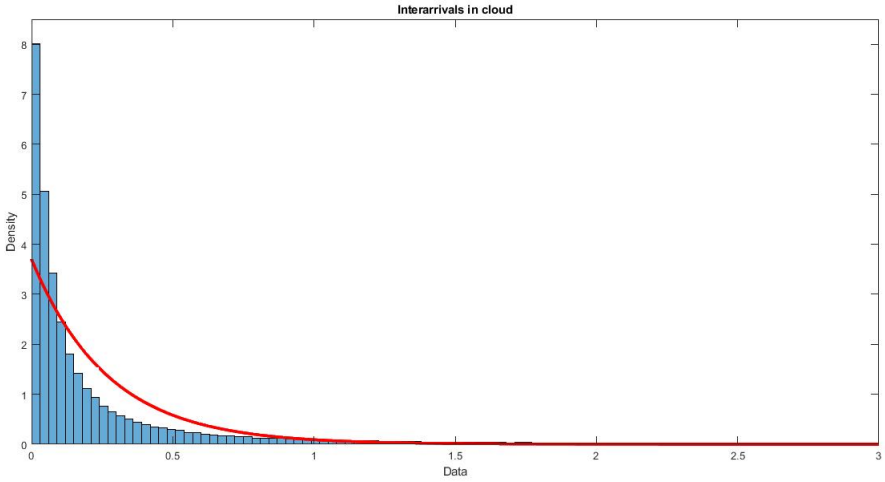


Figura 9.12: Interarrivi al cloud

Conclusioni

Al termine di tutte le simulazioni e relative comparazioni, possiamo affermare che ci troviamo di fronte ad un sistema stazionario.
Veniva richiesto di migliorare l'algoritmo 1 al fine di ridurre il tempo di risposta globale del sistema. Attraverso la prelazione siamo riusciti ad ottenere il risultato richiesto, infatti i task di classe 1 vengono eseguiti principalmente nel cloudlet, e quindi più velocemente, e questo ha permesso di ridurre il tempo di risposta dell'intero sistema. In particolare il miglioramento si ottiene maggiormente aumentando il valore di soglia S il più possibile.