Server Distribuiti con politica di Zoning

Simone Berni

Simulazione di Sistemi

Table of contents

Introduzione

Sistema

Il sistema che si è andato a modellare è un ambiente virtuale di server-client.
Vi è un numero massimo di utenti che è possibile ospitare all'interno: prima di poterne accettare di nuovi è necessario aspettare che altri escano.

Server

All’interno dell’ambiente simulato sono presenti un numero arbitrario di server, che comunicano tra di loro e gestiscono gli utenti durante un ciclo di elaborazione.
Ogni server è formato da due code nelle quali sono inseriti gli utenti. Una coda è attiva e l’altra passiva, e il ruolo viene cambiato ad ogni ciclo di elaborazione.

Coordinatore

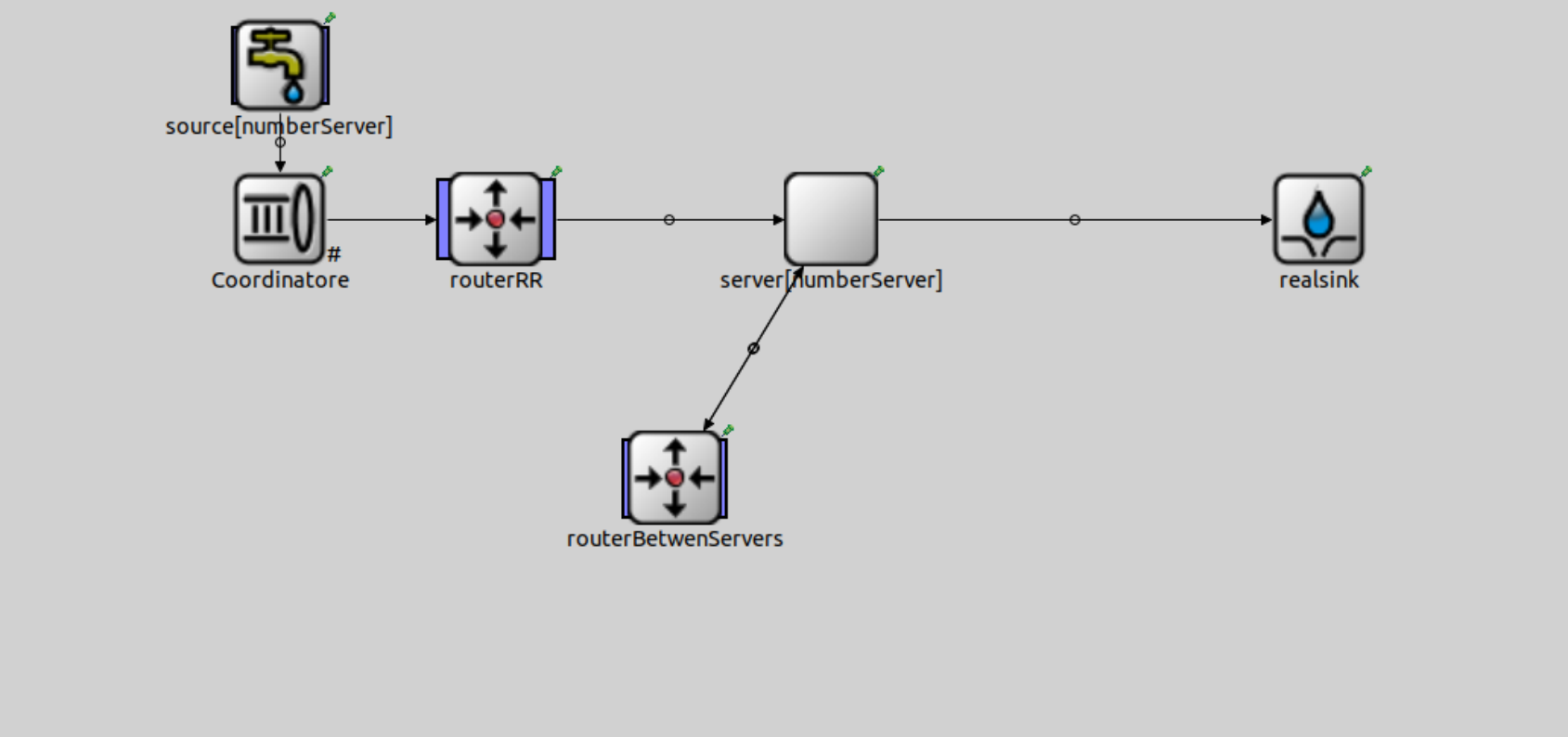
All'interno del sistema vi è una sola entità speciale, il coordinatore che ha il compito di gestire il ciclo di elaborazione e di controllare l'ingresso degli utenti nel sistema.

Utente

Ogni utente all'interno del sistema può svolgere due operazioni: o rimanere nel suo server o cambiare server. Inoltre ogni singolo utente ha un numero massimo di cambiamenti di stato da poter eseguire, una volta terminati deve uscire dal sistema.

Implementazione - Classi

Sistema - Rappresentazione

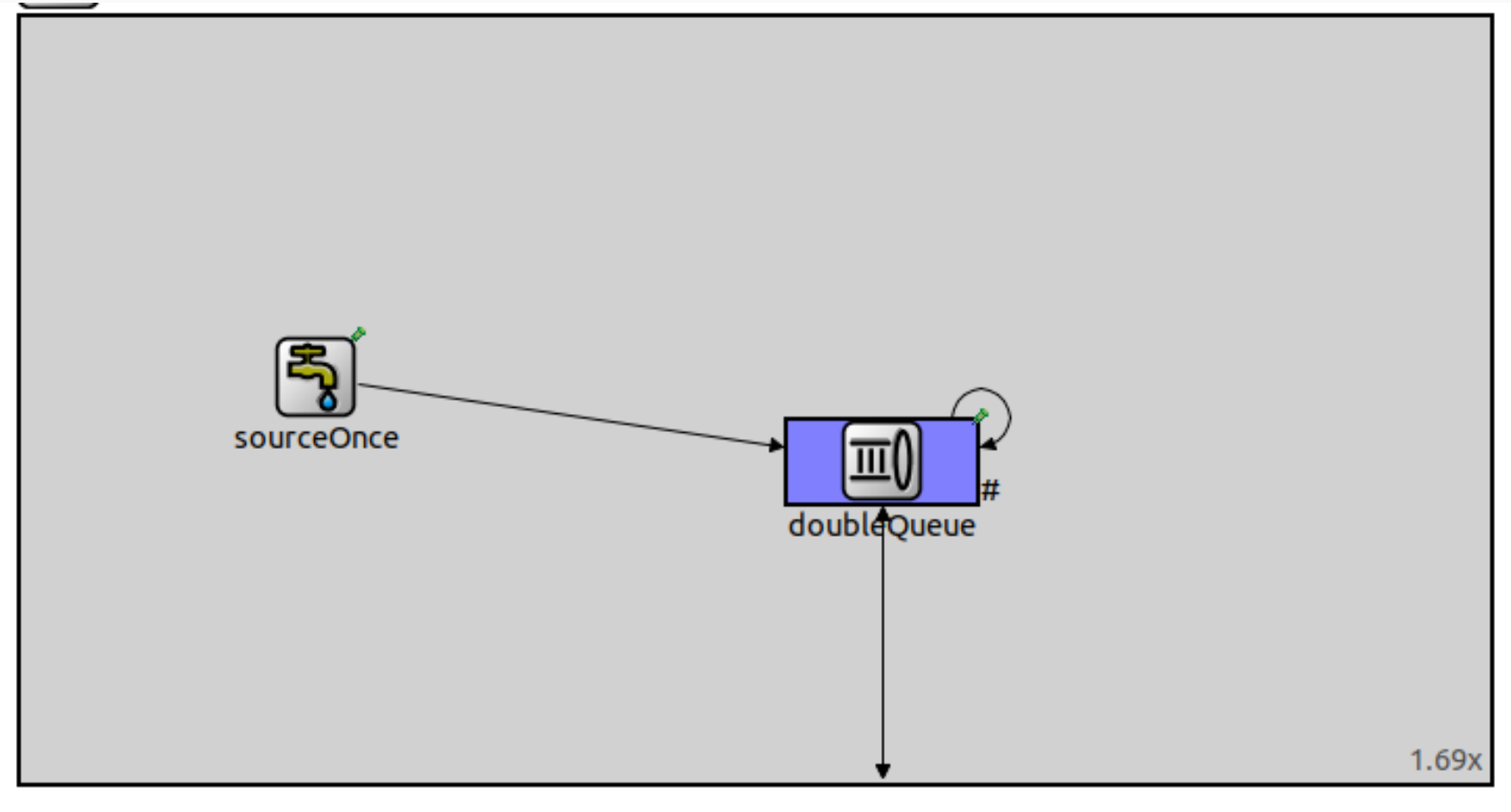


N Sorgenti | 1 Coordinatore | 2 Router RoundRobin | N Server | 1 Sink

Sistema - Funzionamento

- Ogni Sorgente è connessa al Coordinatore e hanno lo stesso interArrivalTime.
- Il Coordinatore controlla se l'utente può entrare e in caso positivo lo invia al RouterRR. In caso negativo è inserito nella coda.
- Il RouterRR utilizzando l'algoritmo RoundRobin invia l'utente arrivato alla coda passiva del server.
- Ogni server gestisce la sua coda attiva e ogni utente o viene inviato nella propria coda passiva, o al RouterBetweenServers o nel Sink.
- Il RouterBetweenServers utilizzando l'algoritmo RoundRobin invia l'utente ad un server diverso da quello mittente.

Server - Rappresentazione



Esterno -> DoubleQueue
SourceOnce -> DoubleQueue
DoubleQueue -> DoubleQueue
DoubleQueue -> Sink
DoubleQueue -> RouterBetweenServers

Server - Caratteristiche

- Modellato a partire dalla nuova classe doubleQueue
- Gestione della coda attiva e di quella passiva
- Aggiunta di una porta @directIn per la ricezione del messaggio InizioElaborazione
- Tre possibili tempi di servizio differenti, scelti a runtime, a seconda che l'utente debba rimanere nello stesso server, cambiare server oppure essere eliminato
- Creazione di nuovi segnali, elaborationTime e zoneLength, con le relative statistiche

Coordinatore

- Simulato da una classe Queue con capienza infinita
- Aggiunta di una porta @directIn per la ricezione del messaggio FineElaborazione
- Aggiunta di attributi come ad esempio maxAvatars e avatarInSystem
- Gestione del ciclo di elaborazione
- Gestione degli ingressi

Implementazione - Dettagli

Ciclo di Elaborazione

La gestione è divisa in due parti:

- Server
  - Quando arriva un Job lo inserisce nella coda passiva.
  - Quando arriva un IE inverte coda attiva con passiva.
  - Quando la coda attiva è vuota, invia un FE al coordinatore.
- Coordinatore
  - All'inizializzazione invia un IE a tutti i server.
  - Quando riceve un FE da parte di un server, aumenta il proprio contatore di uno.
  - Quando ha ricevuto un FE da parte di tutti i server, invia un IE in broadcast.

Vita Utenti

- Ogni utente è rappresentato da un Job
- È stato aggiunto l'attributo time alla classe Job
- È stato realizzato un nuovo costruttore per potergli assegnare il valore alla sua inizializzazione
- Ogni volta che il server elabora un utente, decrementa di uno il valore di time
- Quando time raggiunge zero, il server invia l'utente nel sink

Sorgenti

Problema:

- Ogni server con la propria sorgente
- Coordinatore che monitora l'ingresso nel sistema

Soluzione:

- Sono presenti tante sorgenti quanti server, ma tutte inviano i propri utenti al coordinatore.
- Il coordinatore deciderà se l'utente potrà entrare nel sistema
- L'utente viene smistato attraverso un router RoundRobin

Inizializzazione degli Utenti

Problema:

- Ogni server deve possedere un numero N di utenti all'avvio del sistema

Quasi Soluzione:

- Ogni server ha una propria sorgente monouso che invia all'inizio della simulazione il numero di utenti richiesto
- Non è una soluzione al 100% corretta in quanto l'invio degli utenti richiede necessariamente del tempo e non può essere istantaneo. Limite di Omnet.

Parametri

Ogni valore richiesto nella simulazione è stato inserito come parametro, ottenendo un unico file omnet.ini che gestisce l'intera configurazione del sistema.

```
[General]
network = Project
cpu-time-limit = 300s
real-time-limit = 10s
sim-time-limit = 300s
simtime-resolution = ps
#Project.numberServer = 4 # todo 4 8 10
#Project.maxAvatar = 80 #todo 40 60 80
#Project.source[*].interArrivalTime = exponential(0.166s)
#Project.newServer[*].doubleQueue.serviceTimeDelete=0.15s


Project.maxAvatar = ${40,60,80} #todo 40 60 80
Project.numberServer = ${4,8,10} # todo 4 8 10
Project.source[*].interArrivalTime = exponential(${0.5,0.25,0.166}s)
Project.source[*].timeAliveJob=geometric(0.1)
Project.Coordinatore.capacity=-1
Project.Coordinatore.serviceTime = 0s
Project.newServer[*].doubleQueue.serviceTimeSameServer=uniform(0.01s,0.03s)
Project.newServer[*].doubleQueue.serviceTimeAnotherServer=uniform(0.1s, 0.3s)
Project.newServer[*].doubleQueue.serviceTimeDelete=${0.05s,0.1s,0.15s}
Project.newServer[*].sourceOnce.timeAliveJob = geometric(0.1)
repeat = 20

```

Preprocessing dei dati

Raggruppamento

Per poter analizzare i dati è necessario prima di tutto conglomerare tutti i file che Omnet genera in modo ordinato e coeso.

- Un dizionario che contiene tutti i dati di tutti gli esperimenti
- Ogni chiave della hash table è una tupla, formata dalla configurazione e l'attributo che si sta analizzando.
- Ogni valore di ogni chiave è una matrice, dove ogni riga rappresenta una run

Analisi dei dati

Valori medi

Per ogni matrice nel dizionario, cioè per ogni attributo di ogni configurazione, si crea un array contenente il valore medio di ogni colonna della matrice. Inoltre si va a creare un secondo array contenente le medie prefisse.

Transiente

Il transiente è calcolato partendo dall'array delle medie prefisse. Si cerca il primo indice che ha un valore compreso tra:

- La somma della media dell'array con lo scarto quadratico medio
- La differenza della media dell'array con lo scarto quadratico medio

Tutti gli elementi prima dell'indice così trovato sono stati scartati

Intervalli di Confidenza

L'intervallo di confidenza è calcolato per ogni attributo di ogni configurazione nel seguente modo:

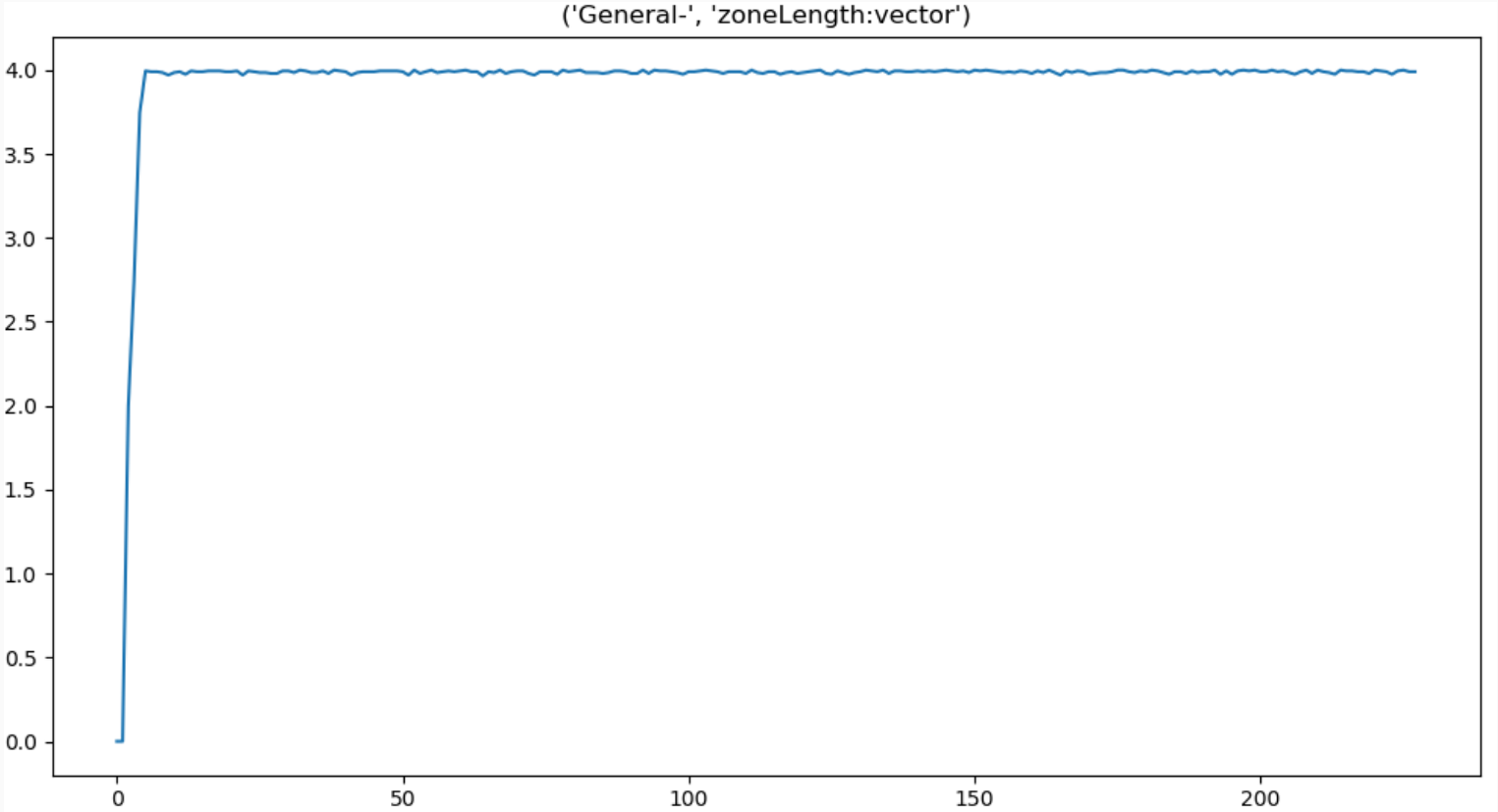
- Per ogni run si calcola il valore medio
- Si calcola il valore medio(X) e lo scarto quadratico mediodi tutte le run(S)
- Si calcola la radice del numero di valori considerati($\sqrt{N}$)
- Si considera l'intervallo di confidenza al 99% e si calcola il valore corrispondente (Z)
- Si ottengono gli intervalli di confidenza nel seguente modo: C= X $\pm$1 Z*S/$\sqrt{N}$

Risultati

maxAvatar=min, numServer=max, interArrivalTime=max, deleteTime=min

Zone Length

Prima del calcolo del transiente



Zone Length

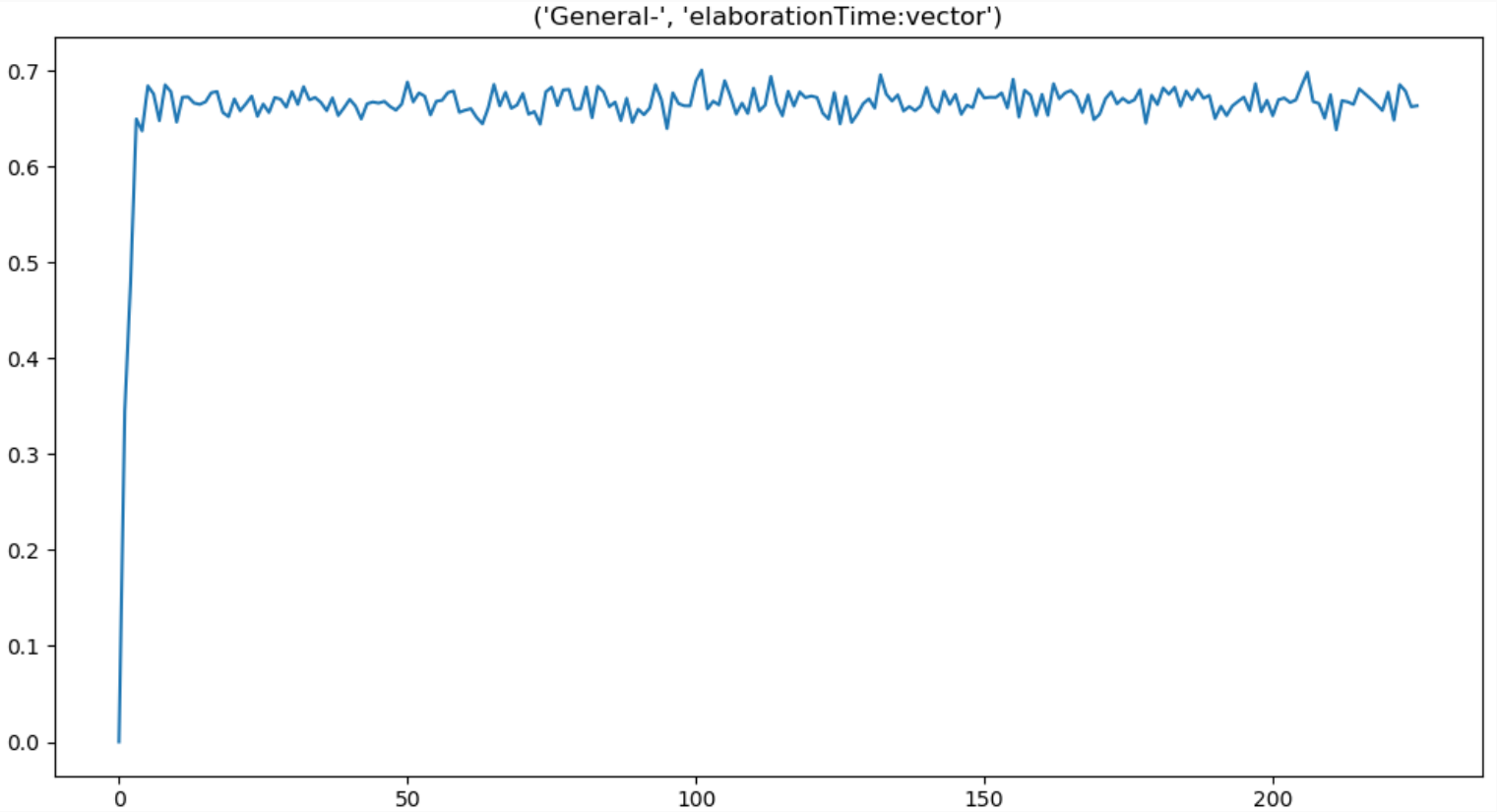
Dopo il calcolo del transiente

Intervallo di confidenza inferiore: 3.984339667911112

Intervallo di confidenza superiore: 3.992074536786341

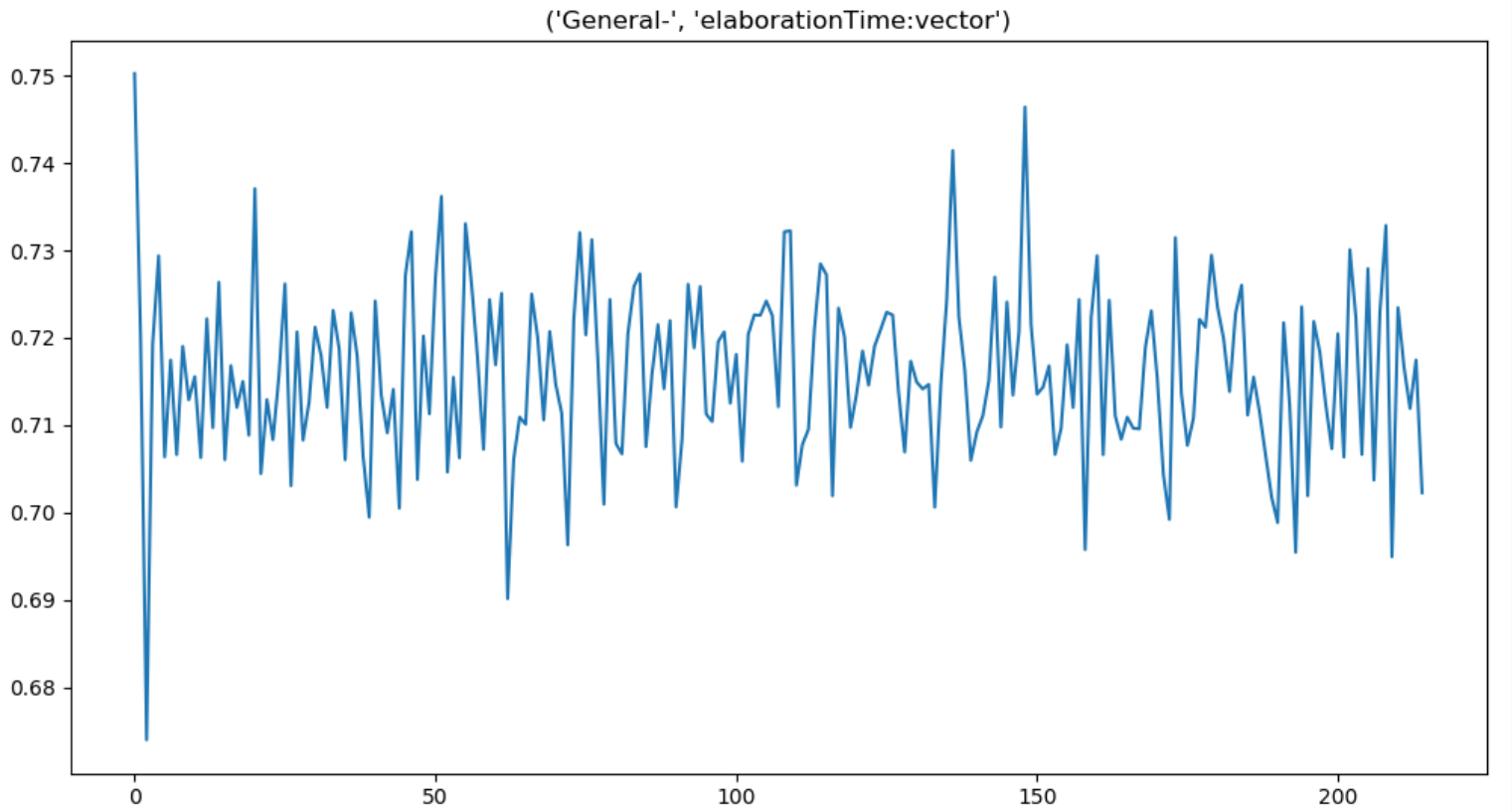
Elaboration Time

Prima del calcolo del transiente



Elaboration Time

Dopo il calcolo del transiente



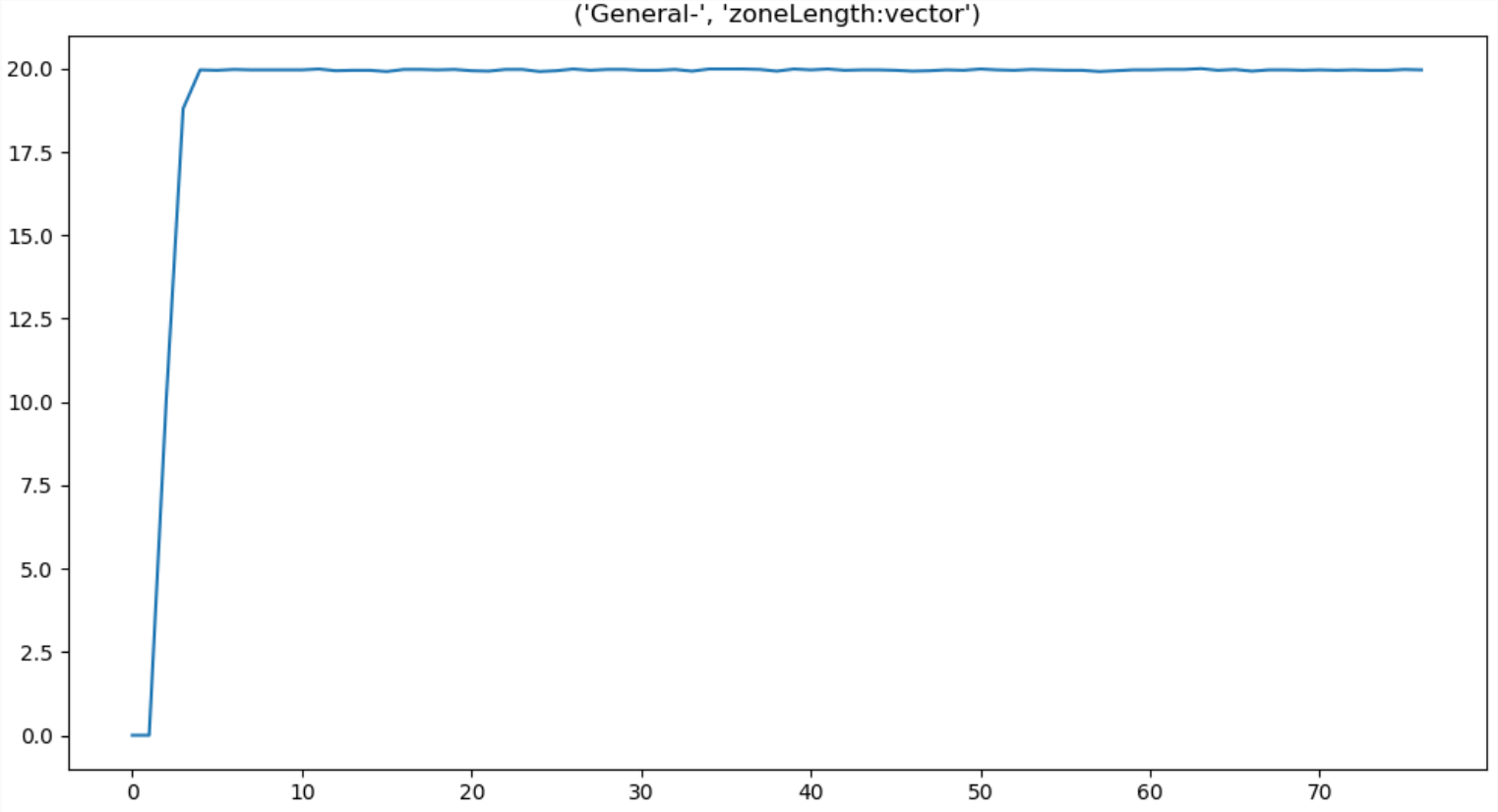
Intervallo di confidenza inferiore: 0.665501401358148 4

Intervallo di confidenza superiore: 0.667419656777814 8

maxAvatar=max, numServer=min, interArrivalTime=min, deleteTime=max

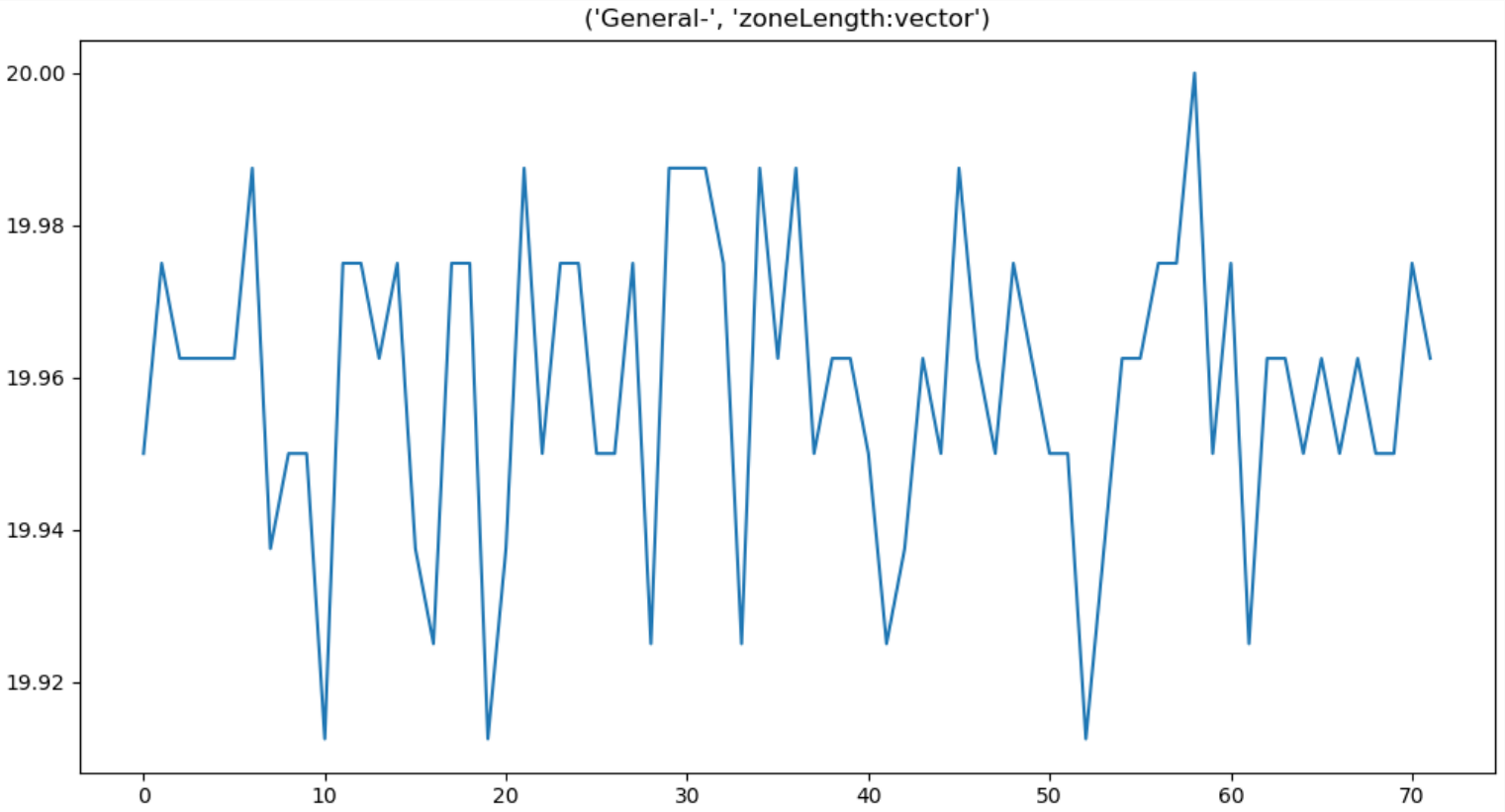
Zone Length

Prima del calcolo del transiente



Zone Length

Dopo il calcolo del transiente

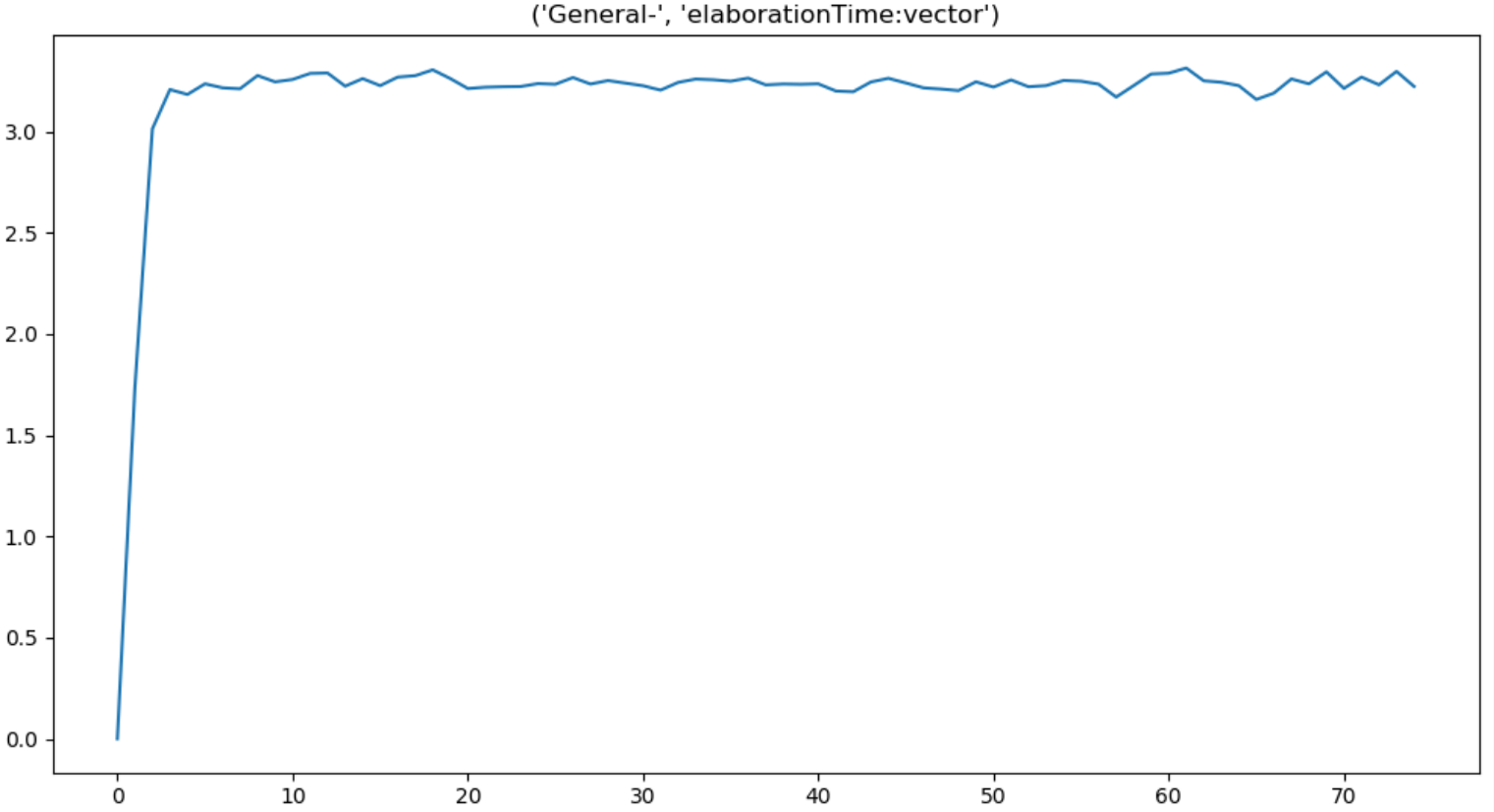


Intervallo di confidenza inferiore: 19.92457122248695,

Intervallo di confidenza superiore: 19.962610922452075

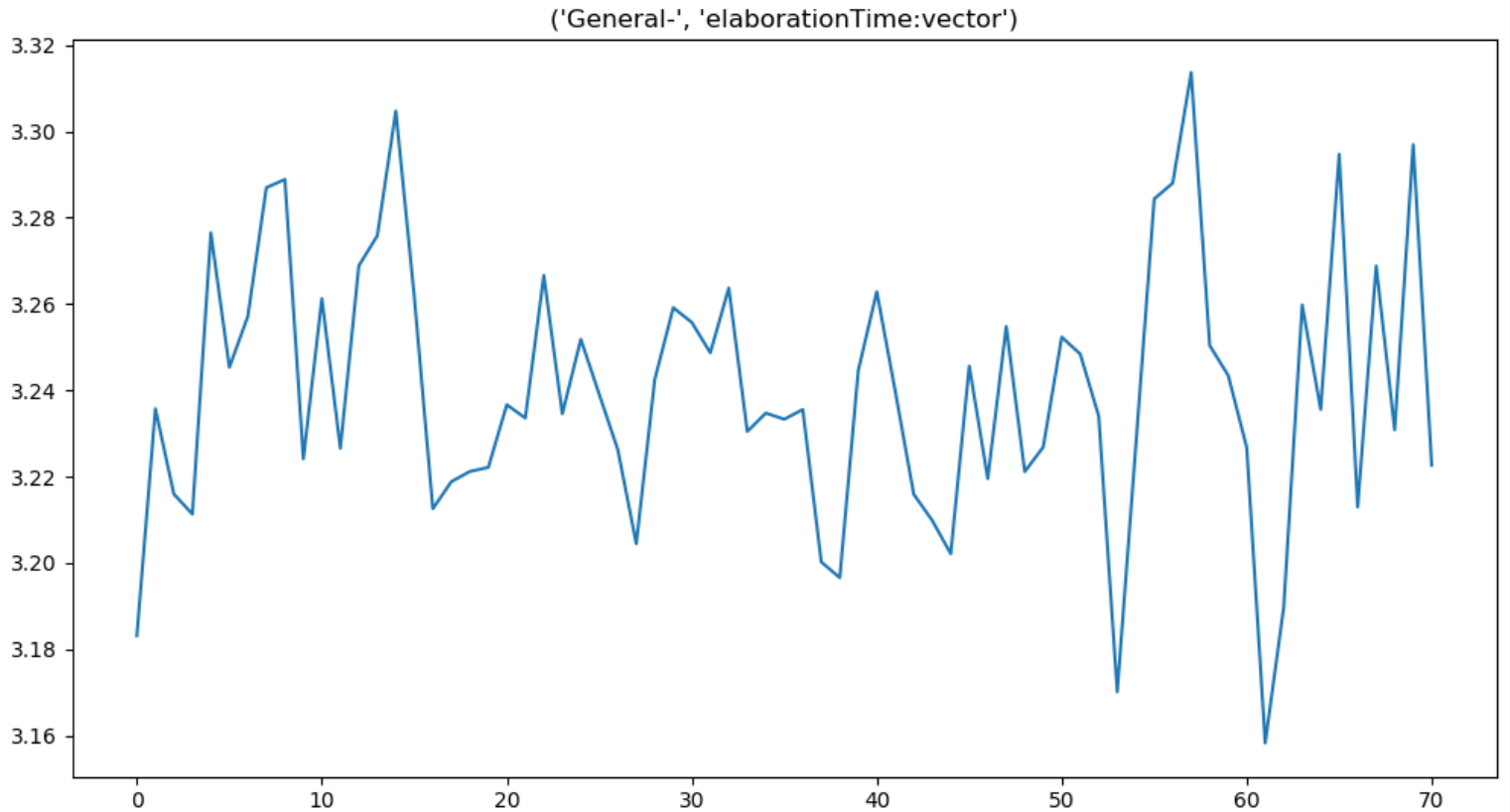
Elaboration Time

Prima del calcolo del transiente



Elaboration Time

Dopo il calcolo del transiente



Intervallo di confidenza inferiore: 3.2307932097894687

Intervallo di confidenza superiore: 3.237999495107467 3