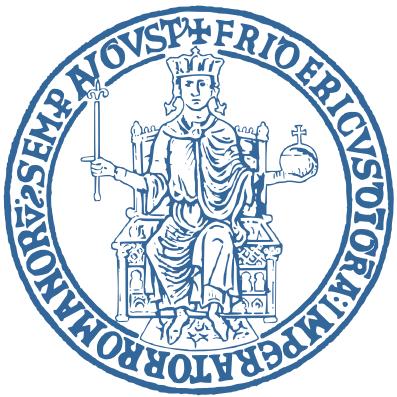


IMPIANTI DI ELABORAZIONE

2023/2024

Relatori:

DOMENICO COTRONEO
PIETRO LIGUORI
STEFANO ROSIELLO



**MAURO GALATEO
CLAUDIO DOTANI
GIUSEPPE FERRARA**



[github.com/mgalateo/Impianti-di-
Elaborazione-2023-2024](https://github.com/mgalateo/Impianti-di-Elaborazione-2023-2024)



Sommario

Introduzione	5
1. Web Server Performance Analysis.....	6
1.1 Fase preliminare	6
1.2 Pianificazione	6
1.3 Settaggio Thread-Group.....	8
1.4 Settaggio Sampler	9
1.5 Dimensione Risorse.....	9
1.6 Esecuzione.....	9
1.7 Plotting dei risultati	10
1.8 Fairness.....	12
1.8.1 Fairness delle risorse	12
1.8.2 Fairness dei singoli thread.....	14
1.9 Bottleneck Performance	16
1.10 Conclusioni	19
2. PCA e Clustering	20
2.1 Raccolta dei dati	20
2.2 Pre-filtraggio dei dati	22
2.3 Analisi delle componenti principali	23
2.4 Clusterizzazione	24
2.5 Conclusioni	25
3. Workload Characterization.....	26
3.2 Raccolta dei dati	27
3.2.1 HL.....	27
3.2.2 Prefiltraggio HL	28
3.2.3 Filtraggio delle richieste.....	31
3.2.4 Workload Sintetico.....	32
3.3 Workload Sintetico di Basso Livello LL'	33
3.3.1 Prefiltraggio LL'	33
3.3.2 PCA Workload Sintetico LL'	33
3.3.3 Clustering	34
3.3.4 Data Validation – Hypothesis Tests	34
4. Design of Experiment.....	43
4.1 Configurazione del piano di DOE.....	43
4.2 Analisi dell'importanza	45
4.3 Analisi della significatività statistica dei fattori	47
4.3.1 Analisi della normalità	47
4.3.2 Analisi omoschedasticità	48

4.3.3 Analisi significatività	49
5. Benchmark.....	50
5.1 Raccolta pre-campione.....	52
5.2 Analisi della normalità delle distribuzioni	54
5.3 Dimensione Campionaria.....	56
5.4 Differenza statistica.....	57
5.5 Risultati e conclusioni	60
6. Regressione.....	62
6.2 Dataset Mail Server 1.....	62
6.2.1 Valutazione delle rette di regressione e R ²	62
6.2.2 Test di Normalità.....	63
6.2.3 Test di Mann-Kendall.....	64
6.2.4 Pendenza e intercetta della retta di regressione.....	64
6.3 Dataset Mail Server 2.....	66
6.3.1 Valutazione delle rette di regressione e R2	66
6.3.2 Test di Normalità	66
6.3.3 Test di Mann-Kendall	67
6.3.4 Pendenza e intercetta della retta di regressione	68
6.4 Confronto tra Mail Server 1 e Mail Server 2	69
6.5 Dataset – OS1	70
6.5.1 Valutazione delle rette di regressione e R ²	70
6.5.2 Test di Normalità	71
6.5.3 Test di Mann-Kendall	72
6.5.4 Pendenza e intercetta della retta di regressione	73
6.6 Dataset – OS2.....	74
6.6.1 Valutazione delle rette di regressione e R ²	74
6.6.2 Test di Normalità.....	75
6.6.3 Test di Mann-Kendall.....	76
6.6.4 Pendenza e intercetta della retta di regressione.....	78
6.7 Dataset – OS3.....	80
6.7.1 Valutazione delle rette di regressione e R ²	80
6.7.2 Test di Normalità.....	81
6.7.3 Test di Mann-Kendall.....	82
6.7.4 Pendenza e intercetta della retta di regressione.....	83
6.7.5 Confronto tra OS1-OS2-OS3	83
6.8 VmRes1 – Failure Prediction.....	84
6.8.1 Valutazione della retta di regressione e R2	84
6.8.2 Test di Normalità.....	84

6.8.3 Test di Mann-Kendall.....	85
6.8.4 Pendenza e intercetta della retta di regressione.....	85
6.8.5 Valutazione Failure Prediction.....	86
6.9 VmRes2 – Failure Prediction.....	87
6.9.1 Valutazione della retta di regressione e R^2	87
6.9.2 Test di Normalità.....	87
6.9.3 Test di Mann-Kendall.....	88
6.9.4 Pendenza e intercetta della retta di regressione.....	88
6.9.5 Valutazione Failure Prediction.....	89
6.10 VmRes3 – Failure Prediction	90
6.10.1 Valutazione delle rette di regressione e R^2	90
6.10.2 Test di Normalità.....	91
6.10.3 Test di Mann-Kendall.....	91
6.10.4 Pendenza e intercetta della retta di regressione.....	92
6.10.5 Valutazione Failure Prediction.....	92
7. Reliability	93
7.1 Esecizio 1	94
7.2 Esercizio 2.....	96
7.3 Esercizio 3.....	98
7.4 Esercizio 4.....	100
7.5 Esercizio 5.....	101
8. Field Failure Data Analysis	105
8.2 Mercury	106
8.2.2 Logging collection e Filtering	107
8.2.3 Data manipulation	109
8.2.4 Data Analysis	111
8.3 Blue Gene/L	115
8.3.1 Logging collection e Filtering	116
8.3.2 Data Manipulation.....	118
8.3.3 Data Analysis	121
8.4 Confronto Reliability tra Mercury e Blue Gene.....	124
8.5 Analisi CWIN tra nodi (Mercury, BG/L) e categorie di errore (Mercury)	126
8.6 Confronto Reliability Mercury Nodi Computazionali	131
8.7 Correlazione tra nodi e categorie di errore nel calcolatore Mercury.....	132

Introduzione

In quest' elaborato verrà affrontata una trattazione riguardante gli Homework assegnati durante il corso di Impianti di Elaborazione a.a. 2023/2024. Nello specifico verranno mostrati i risultati ottenuti in ogni Homework corredati dalle relative analisi, valutazioni e conclusioni.

Gli Homework trattati consistono in:

- ❖ **Web Server**
 - Capacity Test
 - Workload Characterization
 - Hypothesis Test
 - Design of Experiment
- ❖ **PCA & Clustering**
 - Cinebench R23 & Performance Monitor
- ❖ **Benchmark**
 - Nbody
- ❖ **Regression**
- ❖ **Reliability Block Diagram – RBD**
- ❖ **FFDA**



<https://github.com/mgalateo/Impianti-di-Elaborazione-2023-2024>

1. Web Server Performance Analysis

Il capacity test è un tipo di test di performance che si propone di valutare la massima capacità di un sistema, come un server web, di gestire un carico di lavoro elevato senza compromettere le sue prestazioni. Durante questo test, viene effettuato l'invio di un elevato numero di richieste al server entro un breve lasso di tempo, al fine di identificare il punto in cui il server inizia a manifestare segni di sovraccarico e le sue prestazioni iniziano a declinare.

1.1 Fase preliminare

In questa fase vanno identificati gli obiettivi, vengono scelti i tools e le metodologie per la generazione del carico.

In particolare le metriche che interessa valutare in questo caso sono:

- Response Time
- Throughput
- Knee Capacity
- Usable Capacity

1.2 Pianificazione

Lato Client è stato installato il tool JMeter, costruendo così un Test-Plan

Per il settaggio **Thread Group** si è optato per:

- ❖ **N. Threads:** 100 (Utenti Virtuali);
- ❖ **Ramp-up Period:** tempo di attivazione dell' ultimo Thread: settato pari a 10 (Verrà attivato un utente ogni 0,1 sec);
- ❖ **Duration:** 300 sec (5 min)
- ❖ **CTT:** Variabile in base agli esperimenti (2000 – 6000) [Richieste/min]

Si propone dunque il seguente sistema:

Macchina Host



Modello	MacBook Pro (15-inch, 2016)
Processore	Intel Core i7 quad-core 2,6 GHz
Ram	16 GB 2133 MHz LPDDR3
Scheda Grafica	Radeon Pro 450 2 GB
Storage	SSD PCIe NVMe M.2 da 256 GB
Sistema Operativo	macOS Monterey 12.7.1

Client



Versione:
5.6.2

Server



Versione:
2.4.52

Macchina Virtuale



Modello	Parallels Desktop 18.0.1
Processore	1 CPU
Ram	1 GB
Scheda Grafica	1 GB
Sistema Operativo	Ubuntu 22.04

1.3 Settaggio Thread-Group

A questo punto sono state eseguite **3 misurazioni di 5 min l'una** per ogni valore di **Constant Throughput Timer CTT [1000 – 2000 – 3000 – 4000 – 4500 – 5000 - 6000]** che indica il numero di richieste al minuto che ogni Thread-Group può fare.

Thread Group

Name:

Comments:

- Action to be taken after a Sampler error:

Continue Start Next Thread Loop Stop Thread Stop Test Stop Test Now

- Thread Properties

Number of Threads (users):

Ramp-up period (seconds):

Loop Count: Infinite

Same user on each iteration

Delay Thread creation until needed

Specify Thread lifetime

Duration (seconds):

Startup delay (seconds):

1.4 Settaggio Sampler

Viene settato l'ip del server verso il quale i client chiedono le risorse, per lo scopo è stata utilizzata una macchina virtuale con **Linux Ubuntu 22.04** equipaggiato con **1 GB di Ram e 1 Core per la CPU**, successivamente sul sistema è stato installato un server **Apache**.

Nel sampler i parametri settati sono stati:

- ❖ **Ip del server;**
- ❖ **Il Path**, che identifica il nome della risorsa;
- ❖ **Set GET**, in quanto si eseguono richieste;
- ❖ **Set Port number: 80** [Port Numeber per richieste http].

Per rendere il test realistico è stato definito un **Random Controller** che in maniera casuale andrà a gestire le diverse richieste **HttpRequest**. In ultima istanza, si va a definire un **Listener** che permetterà di monitorare l'output del test plan, tra questi ultimi si è scelto il **Simple Data Writer**, che consente di salvare le informazioni di ogni singola richiesta all' interno di un csv.

1.5 Dimensione Risorse

Come dimensione delle risorse che vengono richieste sono state scelte 5 immagini rispettivamente di **[16.6kB – 49.4kB – 100.5kB – 1.1MB – 6.0 MB]** in modo da avere risorse distribuite per carico.

	Nome	Dimensione	Modifica
⌚ Recenti	index.html	10,7 kB	20 nov
★ Preferiti	EXTRASMALL.jpg	16,6 kB	20 ott
🏠 Home	SMALL.jpg	49,4 kB	18 ott
⬇ Documenti	MEDIUM.jpg	100,5 kB	14 ott
⬇ Documents	LARGE.jpg	1,1 MB	5 ott
🎵 Music	EXTRALARGE.jpg	6,0 MB	18 ott
🖼 Pictures			
🎥 Videos			

1.6 Esecuzione

Il test plan è stato lanciato **3 volte** per ogni livello di CTT e tra un esperimento e l'altro viene riavviato il server al fine di evitare fenomeni di accumulo di caching che possono invalidare i risultati. Ogni esperimento ha la durata di 5 minuti ciascuno.

Durante l'esecuzione dei test plan vengono monitorati i parametri delle prestazioni di **Throughput** come:

$$\text{Throughput} = \frac{\text{NumeroRichieste}}{\text{TempoDiEsecuzione}}$$

e **Response Time** come la media dei valori della colonna “Elapsed”.

Per quanto concerne i punti di **Knee Capacity** e **Usable Capacity**, si rende necessario il calcolo della potenza che per definizione è:

$$\text{Power} = \frac{\text{Throughput}}{\text{ResponseTime}}$$

1.7 Plotting dei risultati

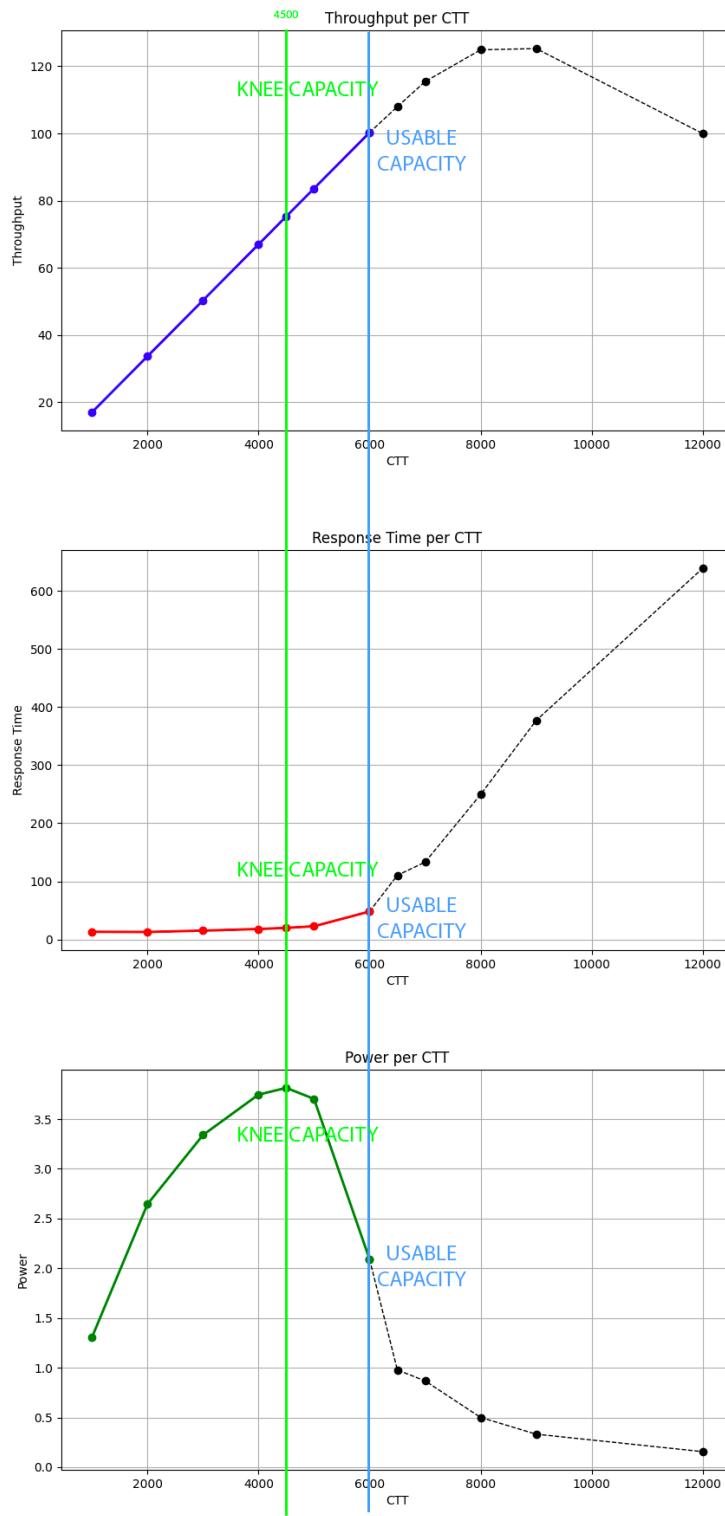
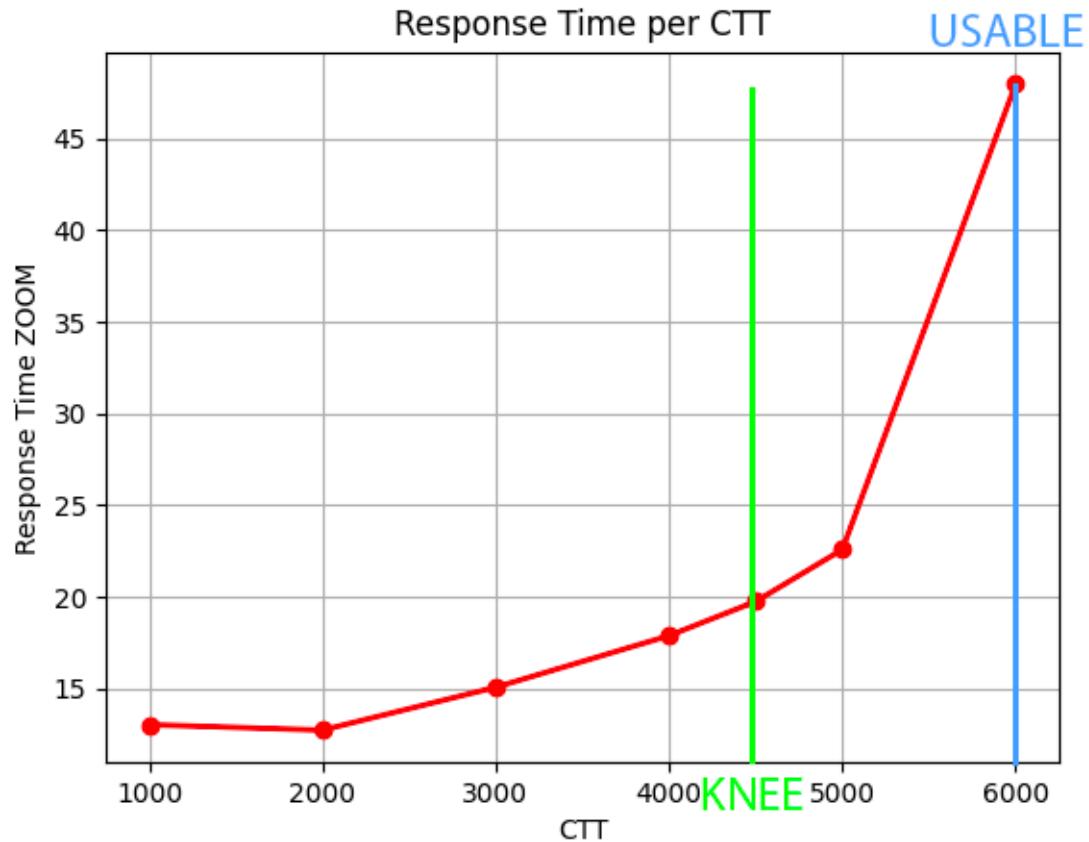


Figura 1.1: Knee Capacity & Usable Capacity

Dal grafico della potenza è possibile andare a stabilire i punti di **Knee Capacity [4500CTT]** e **Usable Capacity [6000 CTT]**, in corrispondenza del picco del grafico della potenza si è scelto il punto di knee capacity, che corrisponde al punto in cui il response time è basso e il throughput è alto, mentre in corrispondenza del grafico power, dove il grafico tende a 0, si posiziona il punto di **Usable Capacity**. Il sistema deve lavorare intorno alla Knee capacity perché deve avere un margine nel caso in cui il carico sia superiore rispetto a quello previsto.



Inoltre Superato il carico di **6000CTT** è stato graficato in nero l'andamento, andando a “simulare” il comportamento del sistema, con carichi più alti di quelli che quest’ultimo riesce a gestire, notando un notevole degrado di performance.

1.8 Fairness

$$Fairness = f(x_1, x_2, \dots, x_n) = \frac{(\sum_{i=1}^N x_i)^2}{n \sum_{i=1}^N (x_i)^2}$$

La valutazione della fairness include l’analisi del throughput per determinare se il sistema sta garantendo un trattamento equo a tutti gli utenti in termini di gestione delle richieste. Se, per esempio, un utente riceve soltanto la metà delle richieste rispetto a un altro utente, ciò potrebbe segnalare un potenziale squilibrio nel sistema in termini di equità.

Dove:

$$x_i = \frac{T_i}{O_i}$$

- ❖ T_i : è la media del Throughput misurato;
- ❖ O_i : è il Throughput fornito per la misurazione;

Le varie fairness in esame sono state calcolate con lo script python fairnessScript.py

1.8.1 Fairness delle risorse

In questo paragrafo vengono analizzate le richieste e viene valutato se queste ultime sono equamente distribuite alle diverse risorse.

Nell’ultima parte dello script fairnessScript.py si è andato quindi a calcolare la fairness per le varie risorse. In prima istanza si è andato a calcolare il thoughtputNormalizzato per ogni esperimento. Tale risultato parziale viene salvato nella cartella fairnessRisorsePerFile che conterrà per ogni esperimento il seguente risultato.

```
1000CTT_1.csv.csv ×
fairnessRisorsePerFile > 1000CTT_1.csv.csv > data
 1  NomeRichiesta,nRichieste,Throughput,ThroughputNorm
 2  ExtraLarge HTTP Request,837,2.7901395069753487,1.0044502225111256
 3  ExtraSmall HTTP Request,825,2.7501375068753435,0.9900495024751238
 4  Large HTTP Request,879,2.9301465073253663,1.054852742637132
 5  Medium HTTP Request,839,2.796806506992016,1.006850342517126
 6  Small HTTP Request,844,2.813474007033685,1.0128506425321266
 7  SuperLarge HTTP Request,875,2.9168125072920312,1.0500525026251313
```

Successivamente viene calcolata la fairness relativa delle risorse per ogni esperimento, elaborando in output il file FairnessRisorse.csv.

Di seguito viene mostrato il risultato ottenuto.

File	CTT	Fairness
1000CTT_1.csv	1000	0.9990565521103947
1000CTT_2.csv	1000	0.9990569242143879
1000CTT_3.csv	1000	0.9990570091109393
2000CTT_1.csv	2000	0.9992429775351376
2000CTT_2.csv	2000	0.9992629885550457
2000CTT_3.csv	2000	0.9992801546714664
3000CTT_1.csv	3000	0.9993324298521012
3000CTT_2.csv	3000	0.9993316270869395
3000CTT_3.csv	3000	0.9993167215073437
4000CTT_1.csv	4000	0.9993540511914987
4000CTT_2.csv	4000	0.9993577902710872
4000CTT_3.csv	4000	0.9993682851058946
4500CTT_1.csv	4500	0.9994006307998649
4500CTT_2.csv	4500	0.9993722006105259
4500CTT_3.csv	4500	0.999408660343891
5000CTT_1.csv	5000	0.9994250266456749
5000CTT_2.csv	5000	0.9993995193767633
5000CTT_3.csv	5000	0.9994171154949768
6000CTT_1.csv	6000	0.9994317925203572
6000CTT_2.csv	6000	0.9994388245285495
6000CTT_3.csv	6000	0.999492281802808
6500CTT_1.csv	6500	0.9996226096274822
6500CTT_2.csv	6500	0.9995401455392212
6500CTT_3.csv	6500	0.9995183968198417
7000CTT_1.csv	7000	0.9993481733453913
7000CTT_2.csv	7000	0.9995645889657707
7000CTT_3.csv	7000	0.9994368577746978
8000CTT_1.csv	8000	0.9993945670579193
8000CTT_2.csv	8000	0.9993743497983608
8000CTT_3.csv	8000	0.9994359845206326
9000CTT_1.csv	9000	0.9990215065409994
9000CTT_2.csv	9000	0.9991364344202409
9000CTT_3.csv	9000	0.9992280156489862

Da tale analisi risulta equa il numero di richieste per risorsa.

1.8.2 Fairness dei singoli thread

Per il calcolo della fairness dei singoli thread si è scelto di procedere in diverse fasi.

Inizialmente viene calcolato la duration di ogni thread per ogni file e tale risultato viene memorizzato nel file DurationXThread.csv. Successivamente viene calcolato il throughput per ogni thread per ogni file e poi normalizzato. Di seguito viene riportato un piccolo estratto del risultato ancora parziale:

Terminata tale fase viene calcolata la fairness per ogni file :

FairnessXThread.csv > data
1000CTT_1.csv,1000,0.9990565521103947
1000CTT_2.csv,1000,0.9990569242143879
1000CTT_3.csv,1000,0.9990570091109393
12000CTT_1.csv,12000,0.9342778714479182
12000CTT_2.csv,12000,0.9811727060424874
12000CTT_3.csv,12000,0.9538369659331412
2000CTT_1.csv,2000,0.9992429775351376
2000CTT_2.csv,2000,0.9992629885550457
2000CTT_3.csv,2000,0.9992801546714664
3000CTT_1.csv,3000,0.9993324298521012
3000CTT_2.csv,3000,0.9993316270869395
3000CTT_3.csv,3000,0.9993167215073437
4000CTT_1.csv,4000,0.9993540511914987
4000CTT_2.csv,4000,0.9993577902710872
4000CTT_3.csv,4000,0.9993682851058946
4500CTT_1.csv,4500,0.9994006307998649
4500CTT_2.csv,4500,0.99937220061805259
4500CTT_3.csv,4500,0.999408660343891
5000CTT_1.csv,5000,0.9994250266456749
5000CTT_2.csv,5000,0.9993995193767633
5000CTT_3.csv,5000,0.9994171154949768
6000CTT_1.csv,6000,0.9994317925203572
6000CTT_2.csv,6000,0.9994388245285495
6000CTT_3.csv,6000,0.999492281802808
6500CTT_1.csv,6500,0.9996226096274822
6500CTT_2.csv,6500,0.9995401455392212
6500CTT_3.csv,6500,0.9995183968198417
7000CTT_1.csv,7000,0.9993481733453913
7000CTT_2.csv,7000,0.9995645889657707
7000CTT_3.csv,7000,0.9994368577746978
8000CTT_1.csv,8000,0.9993945670579193
8000CTT_2.csv,8000,0.9993743497983608
8000CTT_3.csv,8000,0.9994359845206326
9000CTT_1.csv,9000,0.9990215065409994
9000CTT_2.csv,9000,0.9991364344202409
9000CTT_3.csv,9000,0.9992280156489862

In fine viene calcolata la media della fairness per ctt:

CTT	Fairness
1000	0.999056828478574
2000	0.9992620402538832
3000	0.9993269261487949
4000	0.9993600421894935
4500	0.9993938305847605
5000	0.9994138871724717
6000	0.9994542996172383
6500	0.9995603839955151
7000	0.9994498733619532
8000	0.9994016337923043
9000	0.9991286522034089

Anche in questo caso i valori della fairness indicano una ottima equità tra i diversi thread.

1.9 Bottleneck Performance

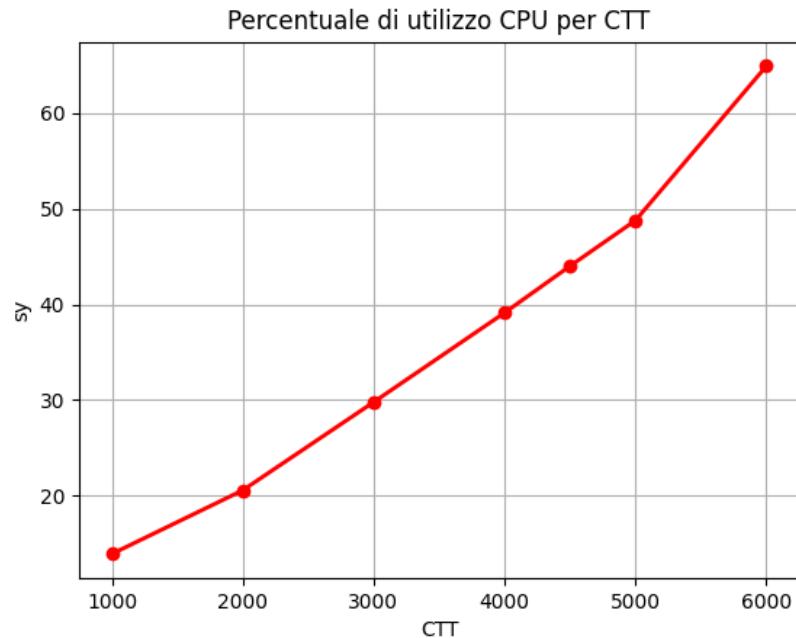


Figura 1.2: Carico x CPU

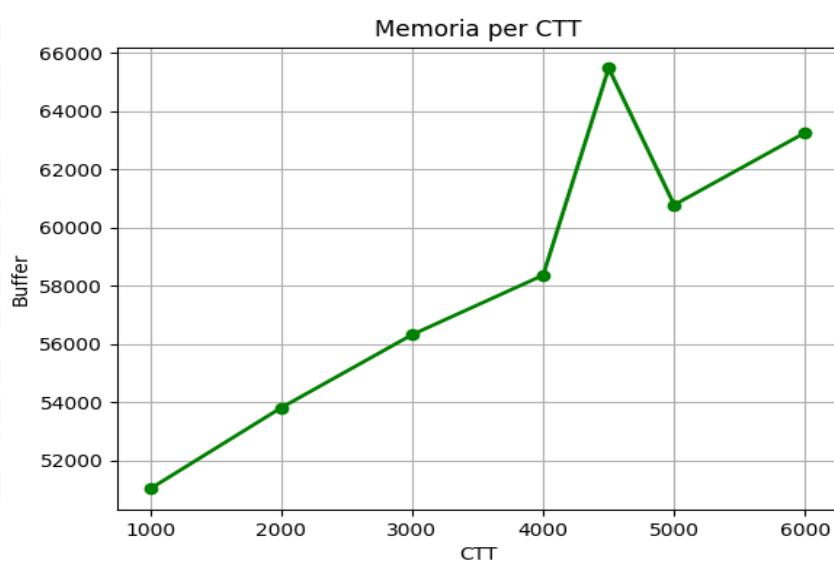


Figura 1.3: Carico x Memoria

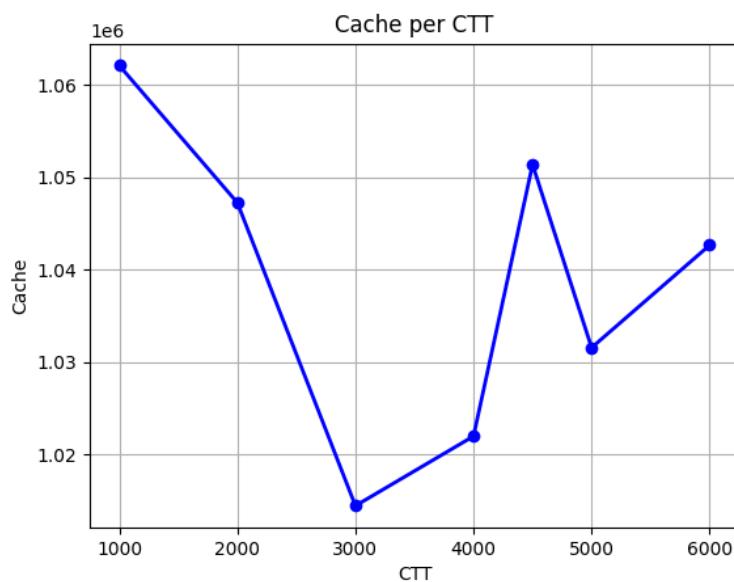


Figura 1.4: Carico x Memoria Cache

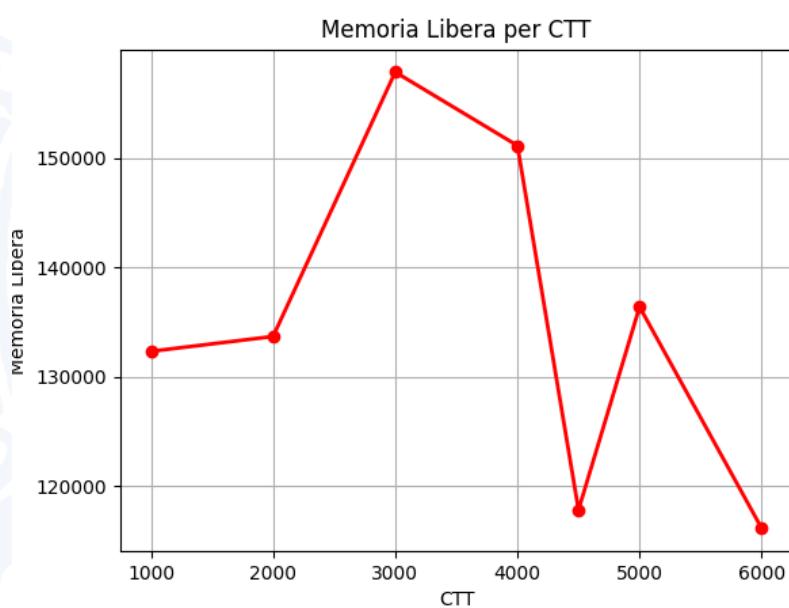
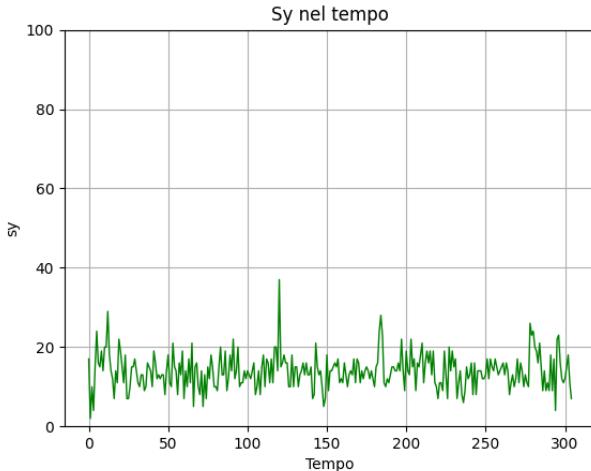
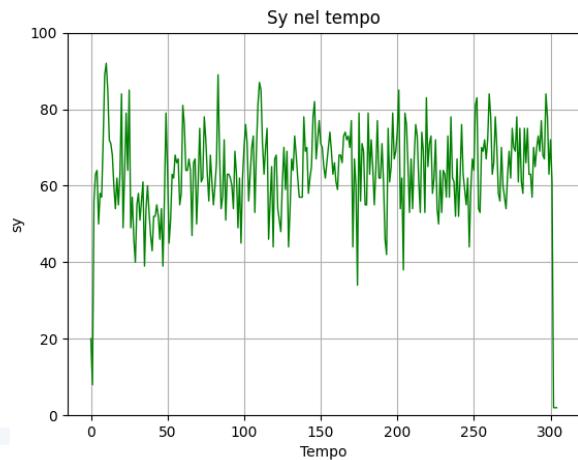


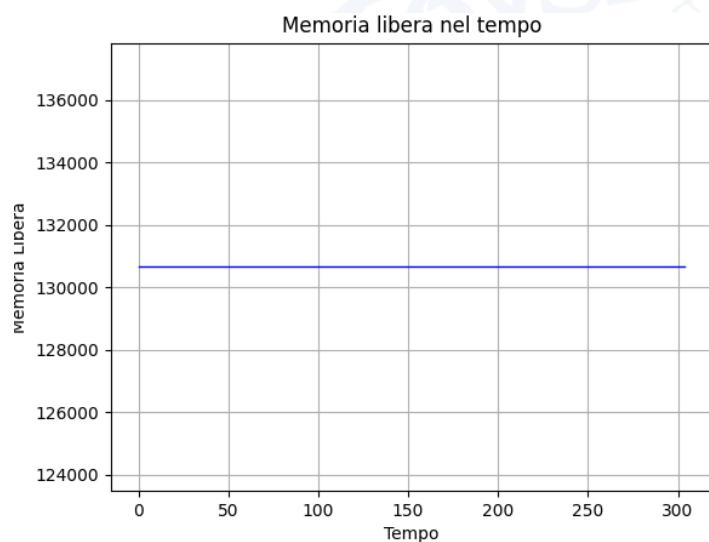
Figura 1.5: Carico x Memoria Libera



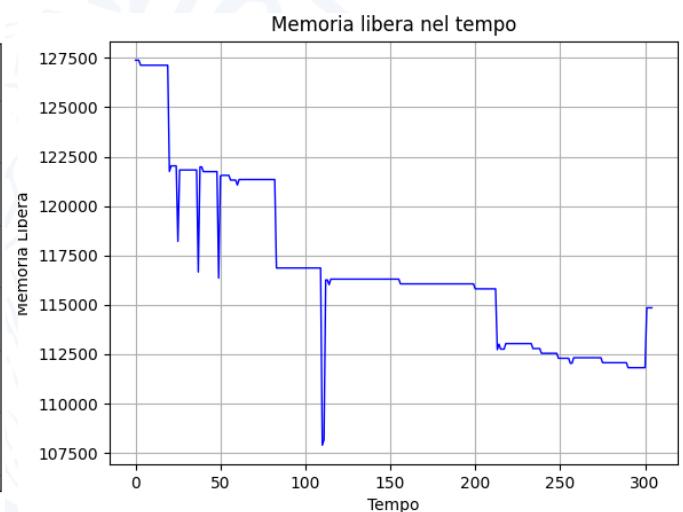
Tempo x CPU Exp: Carico 1000 req/min



Tempo x CPU Exp: Carico 6000 req/min



Tempo x Memoria Exp: Carico 1000 req/min



Tempo x Memoria Exp: Carico 6000 req/min

Da questi confronti si vogliono evidenziare le differenze di carico, durante il singolo esperimento, a basso carico e a alto carico

1.10 Conclusioni

Dai plot si identificano dunque i punti di **Knee Capacity** in corrispondenza del carico di **4500 req/min**. Fino a questo punto il sistema funziona al suo meglio, mostrando incrementi di **throughput** elevati e un basso **response time**. Dopo questo valore, il throughput continua ad aumentare costantemente, ma i tempi di risposta iniziano ad aumentare vertiginosamente, segno del fatto che si sta andando verso saturazione. Pertanto, facendo una scelta conservativa, si è posizionato il punto di **Usable Capacity** a **6000 req/min**. Inoltre è stata garantita la **stabilità** del sistema, ovvero il rapporto dei rispettivi punti è pari al 75%, infatti:

$$\frac{\text{KneeCapacity}}{\text{UsableCapacity}} = \frac{4500}{6000} = 75\%$$

In sostanza, se la Usable Capacity di un sistema è notevolmente superiore alla Knee Capacity, il sistema è più efficiente nel gestire il carico di lavoro. Se si riesce a mantenere il rapporto tra queste due capacità al di sotto o uguale al 75%, possiamo affermare che il sistema è in grado di gestire un volume di richieste più elevato prima che si manifestino problemi di qualità del servizio o di prestazioni.

In altre parole, più grande è la differenza tra la usable capacity e la knee capacity, migliore è la capacità del sistema di gestire il carico di lavoro senza incorrere in problemi. Mantenendo questo rapporto al di sotto del 75%, ci assicuriamo che il sistema possa gestire un alto volume di richieste prima di riscontrare problemi di prestazioni o di qualità del servizio.

Dai grafici del Bottleneck proposti, si evince che il parametro più impattato è la **cpu**, dalla *Figura 1.2* si evince che la cpu arriva al picco di **6000 req/min** prima di saturare, mentre nelle immagini successive non si evince un riscontro oggettivo di bottleneck riguardante gli altri parametri come: **Buffer**, **Memoria Cache** e **Memoria Libera**.

2. PCA e Clustering

Principal Component Analysis, è una tecnica di analisi dei dati utilizzata in statistica e analisi multivariata. Il suo obiettivo è ridurre la dimensionalità di un insieme di dati, preservando al contempo la massima varianza all'interno dei dati stessi.

La PCA aiuta a identificare i modelli principali all'interno di un insieme di dati complessi e a rappresentarli in modo più compatto, riducendo il numero di variabili originali. Questo processo rende più agevole l'analisi dei dati e può anche contribuire a eliminare il rumore o le informazioni meno rilevanti.

Il clustering è una tecnica di analisi dei dati che consiste nel partizionare un insieme di osservazioni o punti dati in sottoinsiemi (cluster) in modo che gli elementi all'interno di ciascun cluster siano più simili tra loro rispetto a quelli in altri cluster, secondo una certa metrica di similarità o dissimilarità. L'obiettivo principale è raggruppare gli elementi in modo tale da massimizzare l'omogeneità all'interno di ciascun cluster e minimizzare l'eterogeneità tra i cluster.

Obiettivo di questo homework è di raccogliere i dati di un sistema sotto stress test e di effettuare pca e clustering sui dati raccolti.

2.1 Raccolta dei dati

Per la raccolta dei dati è stato utilizzato un computer dalle seguenti caratteristiche:

	
Modello	Inspiron 15 Plus
Processore	Intel® Core™ i7-11800H (4,6 GHz)
Ram	16 GB DDR4 a 3.200 MHz
Scheda Grafica	NVIDIA(R) GeForce RTX (TM) 3050 Ti (GDDR6 4GB)
Storage	SSD PCIe NVMe M.2 da 1 TB
Sistema Operativo	Windows 11 Pro 23H2



CINEBENCH

R23

Per mettere sotto stress tale macchina è stato applicato uno stress test chiamato Cinebench R23. Cinebench R23 includono il test di rendering multi-threaded basato su Cinema 4D, un'applicazione di modellazione e animazione 3D. Il benchmark sfrutta la potenza della CPU per valutare le prestazioni complessive del sistema in scenari di carico di lavoro intensivi.

Tale test ha sfruttato tutti gli 8 core del computer.



Performance Monitor

I dati sono stati raccolti utilizzando il tool Windows Performance Monitor.

Si è scelto di monitorare diverse aree di interesse, in particolare:

- **Processore**
 - % Tempo di interrupt
 - % Tempo DPC
 - % Tempo inattività
 - % Tempo privilegiato
 - % Tempo processore
 - % Tempo utente
 - Interrupt/sec
- **Memoria**
 - % Byte vincolati in uso
 - Errore di richiesta nulla/sec
 - Errori cache/sec
 - Errori di pagina/sec
 - Errori in transizione/sec
 - Input pagine/sec
 - Letture pagine/sec
 - MByte disponibili
 - Output pagine/sec
 - Pagine/sec
 - Scritture in copia/sec
 - Scritture pagine/sec
- **Disco Fisico**
 - % Tempo disco
 - % Tempo inattività
 - % Tempo lettura disco
 - % Tempo scrittura disco
 - Letture disco/sec
 - Scritture disco/sec
 - Suddivisione IO/sec
 - Trasferimenti disco/sec

I dati sono stati raccolti nel file DataCollector01.csv il quale contiene 28 colonne e 604 righe.

2.2 Pre-filtraggio dei dati

I primo luogo verifichiamo se ci sono colonne costanti o verifichiamo se esistono correlazioni nei dati.

Eliminiamo la colonna Timestamp perché non di nostro interesse.

Non sono state rilevate colonne costanti quindi procediamo con l'analisi della matrice di correlazione.

Multivariato	Disco fisico/%	Tempo disco	Disco fisico/%	Tempo inattivo	Disco fisico/%	Tempo lettura disco	Disco fisico/%	Tempo scrittura disco	Disco fisico/sec	Lettura disco/sec	Scrittura disco/sec	Disco fisico/Suddivisione IO/sec	Disco fisico/Trasferimento	Disco fisico/Trasferimenti disco/sec
Disco fisico/%	-1,0000		0,7400		0,9994		0,0118		0,8168		0,1927		0,0032	
Disco fisico/%		-0,7409		1,0000		-0,7401		-0,0676		-0,6478		-0,1796		-0,0071
Disco fisico/%			-0,7401		1,0000		0,0447		0,8089		0,1602		-0,0000	
Disco fisico/%				0,1018		-0,0676		1,0000		0,1966		0,5807		0,0561
Disco fisico/Lettura disco/sec		0,8168		-0,0478		0,8089		0,1966	1,0000		0,4630		0,0094	
Disco fisico/Scrittura disco/sec		0,1927		-0,1796		0,1602		0,5807		0,4630	1,0000		0,0301	
Disco fisico/Suddivisione IO/sec		0,0032		0,0071		-0,0000		0,0561		0,0094		0,0301	1,0000	
Disco fisico/Trasferimenti disco/sec		0,6964		-0,5626		0,6778		0,3733		0,9490		0,7376		0,0187
Memoria/% Byte vincolati in uso		-0,0374		0,0473		-0,0425		0,0859		-0,1478		0,0382		0,0406
Memoria/Error di richiesta nulla/sec		0,3379		0,3095		0,3408		-0,0260		0,4357		0,0525		-0,0046
Memoria/Error cache/sec		0,0404		0,0653		0,0376		0,0830		0,0830		0,0760		-0,0038
Memoria/Error di pagina/sec		0,3414		-0,3970		0,3427		0,0028		0,4398		0,0763		-0,0040
Memoria/Error in transizione/sec		0,4471		-0,2916		0,4550		0,0698		0,5195		0,0730		0,0084
Memoria/Input pagina/sec		0,8480		-0,5678		0,8512		0,0667		0,8518		0,1353		0,0173
Memoria/Lettura Pagine/sec		0,0460		0,0450		0,0450		0,0841		0,9513		0,0210		0,0085
Memoria/MB/sec		0,0165		0,1167		0,0453		0,0024		0,0234		0,0238		0,0214
Memoria/Output pagina/sec		-0,0066		0,0300		-0,0071		0,0075		-0,0126		-0,0092		-0,0020
Memoria/Pagine/sec		0,8480		-0,5678		0,8512		0,0067		0,8517		0,1033		0,0073
Memoria/Scrittura in copia/sec		0,1984		-0,1477		0,1924		0,0853		0,2255		0,1097		-0,0003
Processore/% Tempo di interrupt		-0,0446		0,0139		-0,0477		0,0503		-0,0127		0,0331		0,0202
Processore/% Tempo DPC		-0,0098		-0,0383		-0,0164		0,1145		0,0492		0,0822		0,0596
Processore/% Tempo inattivo		0,3043		-0,2928		0,3078		-0,0387		0,4366		0,0410		-0,0062
Processore/% Tempo privilegiato		0,2552		-0,1933		0,2455		0,1867		0,3342		0,1289		0,0472
Processore/% Tempo processore		-0,3085		0,2983		-0,3117		0,0334		-0,4393		-0,0487		0,0065
Processore/% Tempo utente		-0,3021		0,3157		-0,3359		0,0668		-0,4697		-0,0606		0,0020
Processore/Interrupt/sec		-0,0096		0,0665		-0,0129		0,0559		0,0113		0,0960		0,0013

L'immagine completa è disponibile nella repo github.

Evidenziamo le seguenti correlazioni:

- Disco Fisico / Tempo di lettura Disco
- **Disco Fisico / Tempo Disco**

Da cui decidiamo di scartare Tempo di Lettura Disco;

- Disco fisico/Trasferimenti disco/sec
- **Disco fisico/Lettura Disco/sec**
- Memoria/Lettura Pagine/sec

Cancelliamo Disco fisico/Trasferimenti disco/sec e Memoria/Lettura Pagine/sec;

- **Memoria/Errore di richiesta nulla/sec**
- Memoria/Errori di Pagina al/sec

da cui eliminiamo Memoria/Errori di Pagina al/sec;

- **Processore%Tempo processore**
- Processore/% Tempo Utente

Eliminiamo Processore/% Tempo Utente;

- **Memoria/Pagine al secondo**
- Memoria/input Pagine al secondo

Eliminiamo Memoria/input Pagine al secondo;

Siamo quindi riusciti ad eliminare 7 colonne.

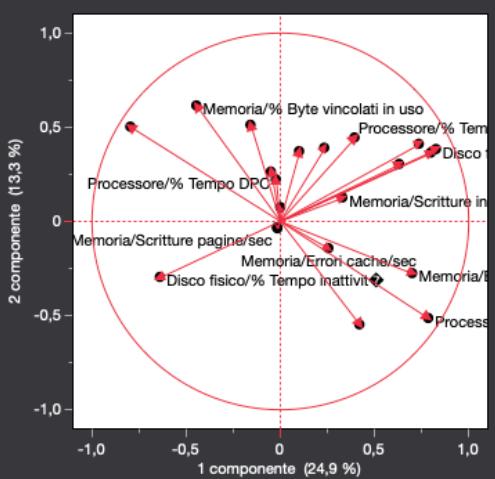
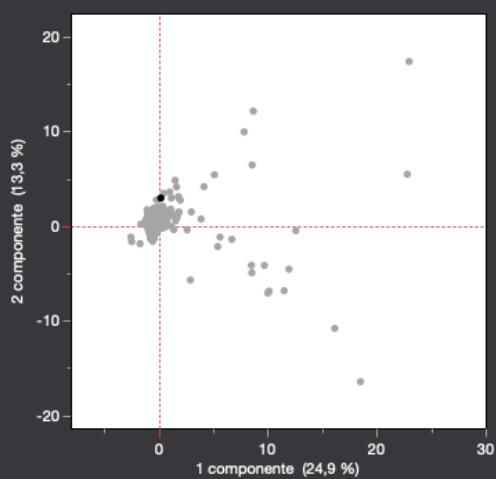
2.3 Analisi delle componenti principali

Componenti principali: sulle correlazioni

Stima della varianza: A livello di riga

▼ Diagrammi di riepilogo

Numero	Autovalore	20	40	60	80
1	5,233290				
2	2,797864				
3	1,869884				
4	1,531478				
5	1,458856				
6	1,362877				
7	1,056945				
8	0,971070				
9	0,889802				
10	0,838168				
11	0,800021				
12	0,549738				



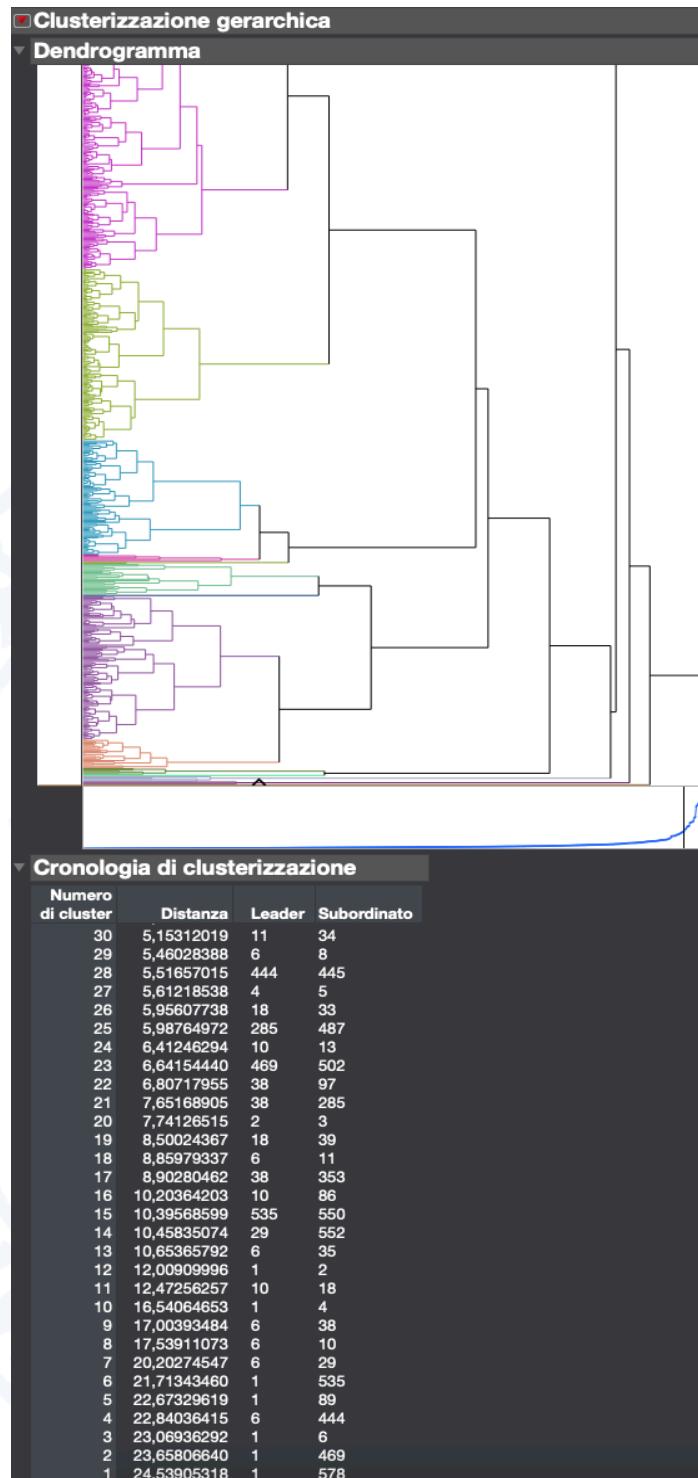
Selezione componente

▼ Autovalori

Numero	Autovalore	Percentuale	20	40	60	80	Percentuale cumulativa
1	5,233290	24,920					24,920
2	2,797864	13,323					38,244
3	1,869884	8,904					47,148
4	1,531478	7,293					54,441
5	1,458856	6,947					61,387
6	1,362877	6,490					67,877
7	1,056945	5,033					72,910
8	0,971070	4,624					77,535
9	0,889802	4,237					81,772
10	0,838168	3,991					85,763
11	0,800021	3,810					89,573
12	0,549738	2,618					92,190
13	0,449050	2,138					94,329
14	0,353049	1,681					96,010
15	0,256646	1,222					97,232
16	0,187487	0,893					98,125
17	0,135392	0,645					98,770
18	0,108297	0,516					99,285
19	0,096059	0,457					99,743
20	0,051961	0,247					99,990
21	0,002066	0,010					100,000

Decidiamo di scegliere 11 componenti principali (salto maggiore).

2.4 Clusterizzazione



In base alle componenti principali ricavate precedentemente, applicando una clusterizzazione gerarchica decidiamo di selezionare 22 cluster (seguendo la regola del gomito).

2.5 Conclusioni

Applicando lo script Matlab Deviance.m verifichiamo la devianza persa.

```

ans          0
B           9.6962e+03
centroid    1x11 double
cluster_data 604x1 double
data         604x21 double
data_norm   604x21 double
DEV_LOST_per 0.2343
DEV_PCA     1.1343e+04
DEV_PCA_CL... 0.7657
DEV_PCA_per 0.8957
DEV_TOT     12663
fid          3
i            19
index       604
N_cluster   22
n_ele        1
N_pca       11
pca_data   604x11 double
var_name    'pca_data'
var_value   604x11 double
W           1.6463e+03
workspace_vars 19x1 struct

```

Dallo script risulta una devianza persa di circa 23%, accettabile per questo contesto.

Il dataset risultante, DatiFinali.xlsx sarà quindi formato da 21 colonne x 22 righe.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	io/%Tempo	%Tempo	%Tempo	sco/Lettura	io/Scritture	/Suddivisi	k Byte	vincere di richiesta/	Errori	carri	in trana/MByte	di/Output	priorità/Pagine	Scrittura	in/%Scrittura	p/%Tempo	core/%Tempo	/%Tempo	/%Tempo	/%Tempo	core/Interru
2	9,715336	91,50194	0,005975	430,2654	1,99197	0	61,17423	58567,89	3667,216	14887,98	7135	0	2087,584	0	0	0	84,50512	0,97246	8,395007	15663,85	
3	3,950815	96,02376	0,143607	325,529	28,86955	0	64,97731	247678,8	43616,91	11398,49	6471	0	2068,652	4,977508	0	0	0	72,97319	2,916411	12,02138	41028,6
4	4,138135	95,895684	0,118849	313,5175	27,21856	0	66,19616	155719,4	12582,03	6428,621	6230	0	1396,211	0	0	0	0	68,38995	3,93803	15,23391	84810,02
5	3,237066	96,92004	0,144207	244,3098	22,65776	0	69,7542	323810,9	4794,579	5582,675	5384	0	1243,221	0	0	0	48,69594	3,559884	36,30692	36964,66	
6	72,3148	47,27702	0,013309	1206,391	5,977163	0	70,45203	242426,8	2433,702	4369,306	5547	0	5615,545	179,3149	0	0	0	48,16851	1,460311	39,44562	36136,93
7	71,58235	60,12743	0,31448	1268,039	85,39655	0	69,64729	32803,2	3878,592	28652,53	5614	0	10042,04	1617,569	0	0	0,096978	0	9,988391	100	56234,62
8	6,828571	93,07632	0,572945	290,7415	147,8347	0	67,43593	42030,39	2841,383	14000,93	6225	0	1092,991	9276,133	0	0	0,192509	29,34692	3,850239	58,32117	42595,11
9	1,690926	98,28641	0,052401	136,0017	22,00028	0	70,01066	139077,8	2365,03	6344,08	5315	0	573,0072	0	0	0	16,69194	5,957098	77,92948	48177,61	
10	0,387614	99,72384	0,347306	2,982108	8,946323	0	69,23256	3950,299	2306,163	2008,947	5385	0	13,9165	0	0	0,194155	0,194145	0	1,941471	100	49911,54
11	0,071589	99,9019	0,071589	0	2,024254	0	69,25093	3550,542	2341,05	849,1747	5760	0	0	7,08489	0	0,098846	0	0	1,28495	100	48171,18
12	0,299684	99,67538	0,216008	0,988124	10,86936	0	69,2554	4153,084	2277,625	1396,219	5756	0	0	0	0	0,192994	0,385998	0	2,798448	100	48560,35
13	0,140185	99,8328	0,140185	0	3,016385	0	69,23489	3305,958	2321,611	407,212	5626	0	0	0	0	0,196392	0,687352	0	2,651193	100	44087,48
14	0,822977	99,15561	0,137941	32,66573	3,959482	0	65,98048	6006,535	2302,439	1573,894	6365	0	149,4705	0	0	0	0,193336	7,49481	1,256698	90,42975	41033,1
15	1,863032	134,1196	0,42754	130,6329	25,31645	0	69,25565	8504,3	4987,34	5069,365	5381	0	2256,202	15,18987	0	0	0	0	2,670131	100	50381,75
16	4,33039	96,20364	0,29506	309,9497	62,38605	0	69,87717	25136,63	3134,156	30451,32	5628	0	3080,682	1004,118	0	0,676914	0,193403	0	12,37792	100	67261,07
17	0,975024	98,99559	0,139056	66,53581	3,972287	0	69,51172	4928,616	2350,601	3024,897	5318	0	313,8107	0	0	0,387914	0	0	1,648618	100	49352,69
18	0,209117	99,76318	0,209117	0	3,996743	0	69,2524	3733,958	2335,097	1376,878	5758	0	7,993487	0	0	0,487877	0,097569	0	1,463641	100	48286,66
19	3,005363	98,42354	0,973961	129,7336	358,7945	0	69,49326	3883,9	2345,34	1410,853	5326	0	19,25733	0	0	0,098977	0	0	1,484751	100	49854,19
20	15,38146	84,99274	2,591195	1212,856	1056,229	0	69,71668	4322,307	3662,666	1490,97	5603	0	868,4775	0	0	0,196104	0,294137	0	3,627791	100	50823,5
21	5,845685	93,24717	0,413091	230,4952	81,82082	0	68,16366	146385,4	109370,5	3543,241	5948	0	1067,662	0,997815	0	0,09745	0	47,15171	5,360073	28,66225	26973,93
22	0,215715	99,75521	0,215715	0	3,996057	0	69,34847	4265,791	2307,723	1361,657	5661	68,93199	68,93199	0	0,999014	0	0,097553	0	1,756066	100	44012,58
23	1,169971	98,8032	0,558809	52,67104	56,64621	1,987586	69,99076	6007,48	2435,787	2281,749	5546	0	239,5042	31,80138	0	0,194097	0,291151	0	3,493794	100	45436,22

3. Workload Characterization

Il processo di **Workload Characterization** si propone di generare un carico di lavoro sintetico per un particolare caso di test, partendo da un carico di lavoro effettivo. Le metodologie di analisi dati utilizzate durante il processo di **Workload Characterization** includono la **Principal Component Analysis (PCA)** e il **clustering**. In breve, la PCA è una tecnica di riduzione della dimensionalità, che si propone di trasformare un insieme di dati multidimensionale in un insieme di dati a dimensione inferiore, conservando la maggior parte delle informazioni presenti nei dati originali. Dall'altro lato, il clustering è una tecnica che mira a raggruppare gli elementi in base alle loro "similitudini", generando così dei cluster.

- ❖ **Raccolta dei dati:** consiste nella raccolta dei dati sull' utilizzo delle risorse del sistema in esame o dell' applicazione.
- ❖ **Pulizia e preparazione dei dati:** (Preprocessing dei dati): include una possibile eliminazione di dati mancanti o errati, la normalizzazione dei dati, la selezione delle caratteristiche rilevanti, lo studio degli outlier.
- ❖ **PCA:** Si esegue PCA per ridurre la dimensionalità dei dati.
- ❖ **Clustering:** si utilizza il clustering per raggruppare gli elementi dei dati in base alle loro similitudini.
- ❖ **Analisi dei risultati:** sarà necessario analizzare i risultati ottenuti dalla PCA e dal clustering per comprendere le caratteristiche del workload e identificare eventuali problemi o opportunità di ottimizzazione.
- ❖ **Reporting:** è importante presentare i risultati dell' analisi in modo da poter prendere decisioni informate su come gestire il workload.



3.2 Raccolta dei dati

3.2.1 HL

La fase iniziale, prevede la raccolta dei dati di alto livello (HL - Jmeter) e basso livello (LL – Server Apache).

Per la raccolta dei dati HL si è proceduto configurando Jmeter come segue:

4 Thread-Group:

- ❖ TG1 25 thread, RU 10s, duration 75s, CTT 500;
- ❖ TG2 50 thread, RU 20s, duration 75s, CTT 1000;
- ❖ TG3 75 thread, RU 10s, duration 75s, CTT 1500;
- ❖ TG4 100 thread, RU 10s, duration 75s, CTT 2000

Sono state utilizzate 11 risorse per TG con i seguenti tagli di memoria.

Nome	Dimensione
ISCHIA.jpeg	6,0 kB
index.html	10,9 kB
EXTRASMALL.jpg	16,6 kB
immagine.jpg	39,2 kB
SMALL.jpg	49,4 kB
onda.webp	79,1 kB
MEDIUM.jpg	100,5 kB
nasa.jpg	121,1 kB
LARGE.jpg	1,1 MB
test.jpg	4,0 MB
EXTRALARGE.jpg	6,0 MB
ultralarge.jpg	10,2 MB

Le colonne dei dati di alto livello in esame sono:

- ❖ timeStamp
- ❖ elapsed
- ❖ responseCode
- ❖ responseMessage
- ❖ threadName
- ❖ dataType
- ❖ success
- ❖ failureMessage
- ❖ bytes
- ❖ sentBytes
- ❖ grpThreads
- ❖ allThreads

- ❖ URL
- ❖ Latency
- ❖ IdleTime
- ❖ Connect

3.2.2 Prefiltraggio HL

È stato quindi eseguito un prefiltro eliminando le colonne costanti quali:

- ❖ `responseCode`: (valore costante 200)
- ❖ `responseMessage`: (valore costante OK)
- ❖ `dataType`: (valore costante bin)
- ❖ `success`: (valore costante true)
- ❖ `failure message`: (valore costante vuota)
- ❖ `idleTime` : (valore costante 0)

A questo punto è stata analizzata la correlazione tra le colonne grazie alla matrice delle correlazioni in JMP, che ha evidenziato una forte correlazione tra le colonne **grpThreads** e **allThreads**, (con coefficiente di correlazione 1), nonché le colonne **latency** e **connect**, delle quali è stato cancellato connect. Si è proceduto pertanto ad eliminare le colonne **grpThreads** e **connect** come ultima fase di filtraggio.

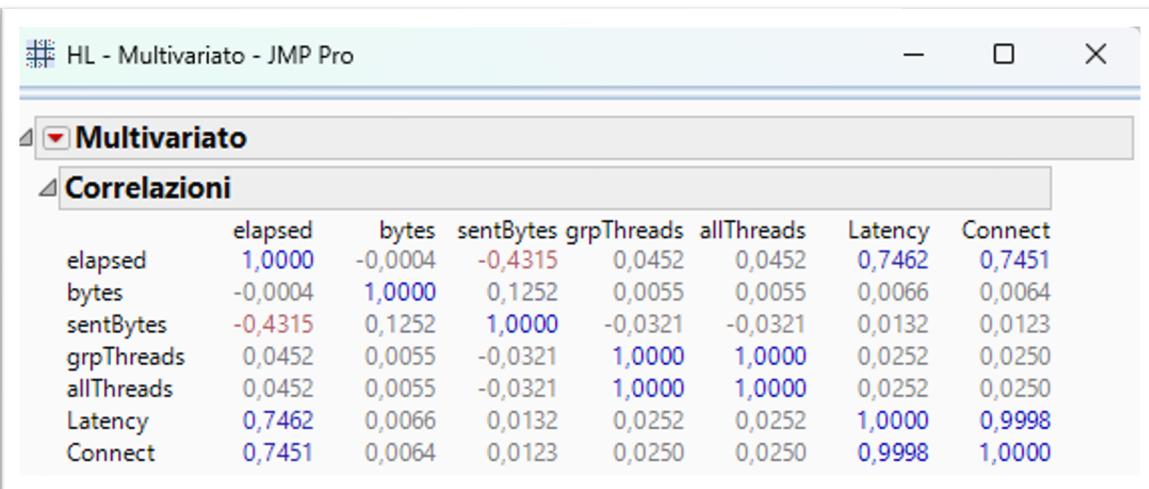


Figura 3.1: Analisi delle correlazioni

A questo punto è stata eseguita la PCA con questi risultati

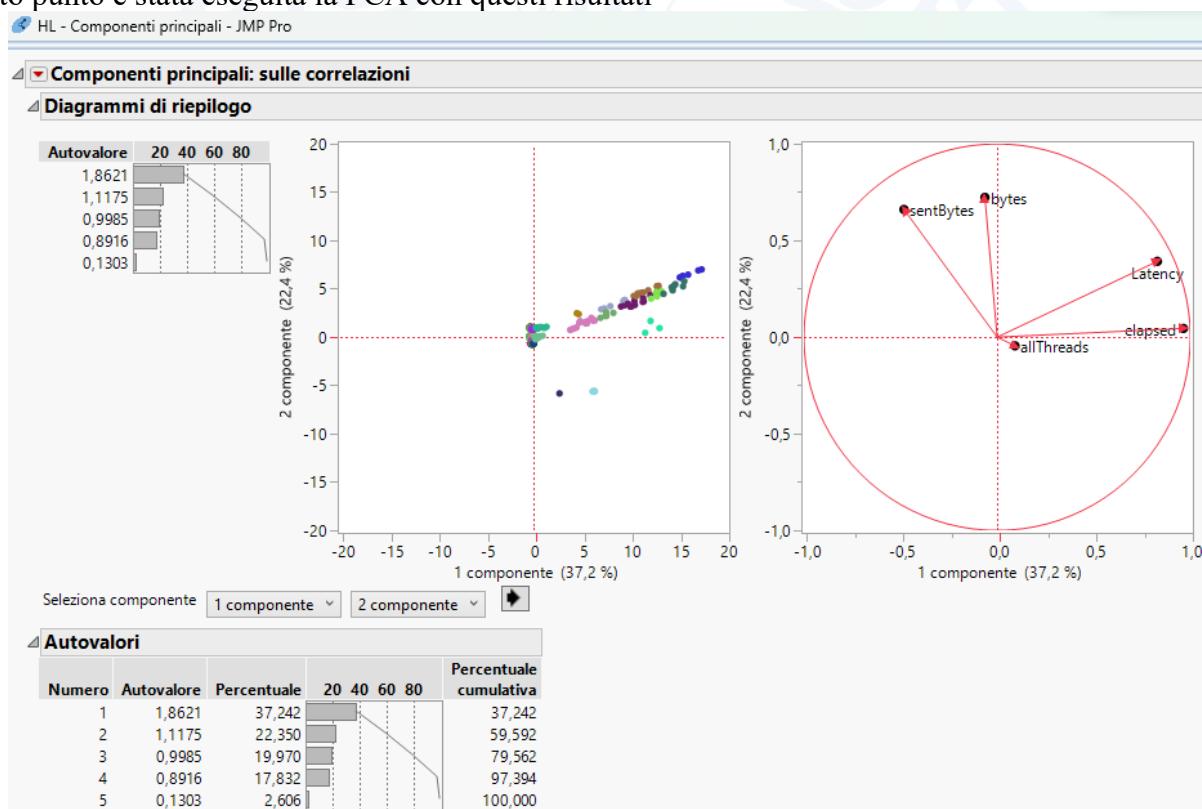


Figura 3.2: PCA HL

Sono quindi state scelte 4 componenti che spiegano il 97,39% della varianza, inoltre ora si procede alla scelta dei cluster.

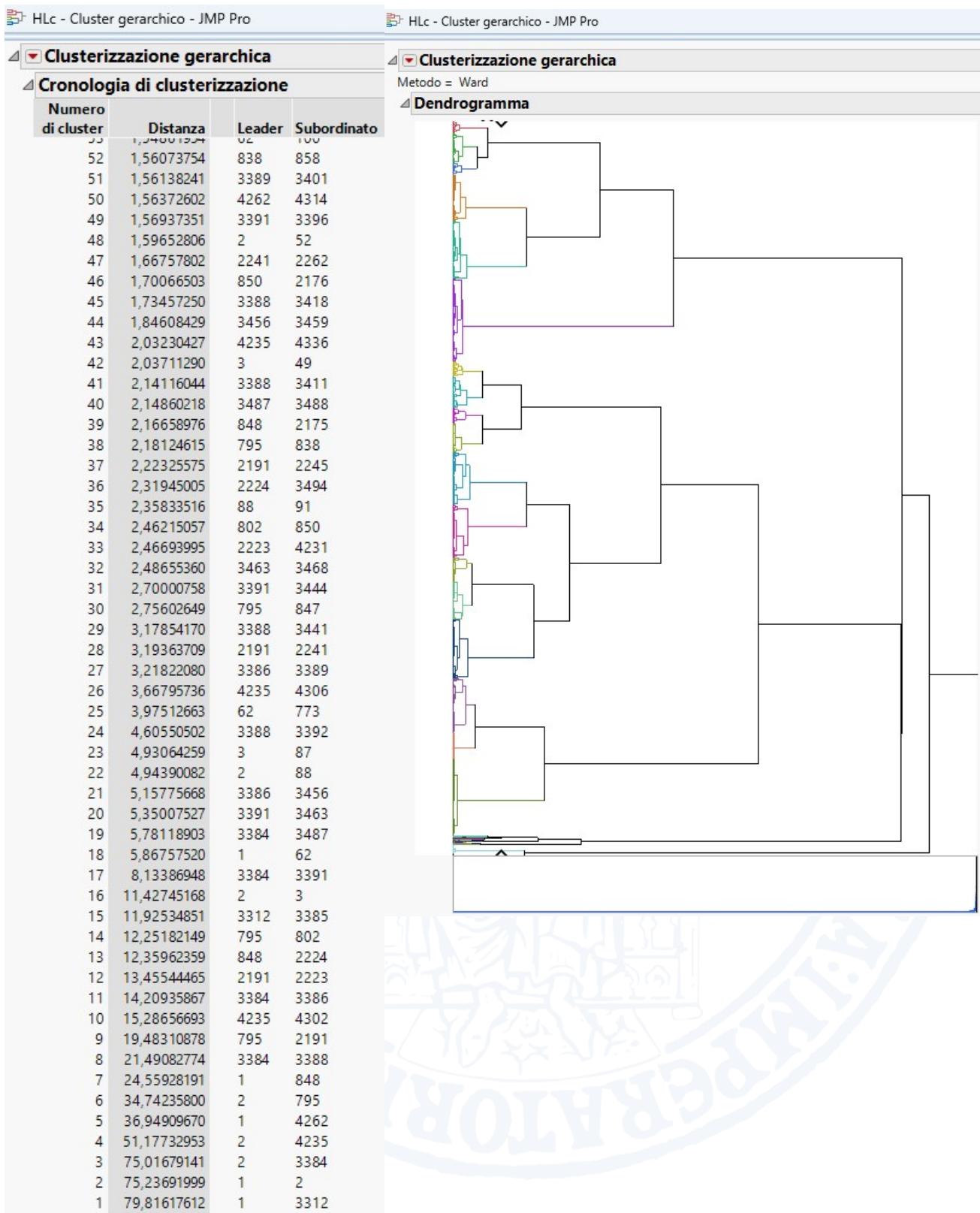


Figura 3.3: clustering

Si scelgono quindi in questo caso 18 cluster, in quanto si nota che dopo 18 cluster non vi sono significative variazioni di distanza.

Per confermare la tesi suddetta, si è scelto di applicare lo script deviance.m per controllare la devianza persa con le scelte effettuate, da cui è scaturito questo risultato:

DEV LOST = 4%

Quindi si evince che la scelta del cluster è stata corretta, in quanto si è presentato un risultato di devianza persa più che accettabile.

3.2.3 Filtraggio delle richieste

Figura 3.4: Richieste da filtrare

A questo punto si è costituita la tabella delle richieste, dalla quale si andranno ad eliminare le colonne non rappresentative del workload.

Cluster	R1_3	R1_4	R1_8	R2_3	R2_5	R2_7	R3_1	R3_2	R3_3	R3_5	R3_6	R3_8	R3_9	R4_2	R4_6	R4_9	
1	0	0	0	75	0	0	25	0	0	0	0	0	25	22	0	0	26
2	0	0	0	0	0	0	121	0	0	0	0	0	13	13	0	0	12
3	0	0	0	0	0	0	0	0	0	0	0	0	165	171	0	0	6
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	275
5	0	78	0	0	24	0	0	0	0	25	21	0	0	0	0	18	0
6	77	0	0	20	0	0	19	14	23	0	0	0	0	0	22	0	0
7	0	0	0	0	143	0	0	0	0	10	11	0	0	0	0	14	0
8	0	0	0	132	0	0	17	16	21	0	0	0	0	0	16	0	0
9	0	0	0	0	0	0	0	0	0	155	177	0	0	0	0	21	0
10	0	0	0	0	0	0	156	143	164	0	0	0	0	0	25	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	243	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	235	0	0
13	0	0	0	0	0	0	0	0	0	0	1	3	2	0	0	0	0
14	0	0	0	0	0	0	0	4	2	2	4	3	4	0	0	0	0
15	0	0	0	0	0	0	0	2	3	4	4	2	1	0	0	0	0
16	0	0	0	0	0	0	7	2	5	1	2	0	2	0	0	0	0
17	0	0	0	0	0	0	10	12	4	10	4	5	13	0	0	0	0
18	0	0	0	0	0	0	3	4	3	4	5	4	1	0	0	0	0

Figura 3.5: Richieste filtrate

3.2.4 Workload Sintetico

A questo punto, a partire dalle colonne filtrate, andiamo a costruire un workload sintetico, composto dalle sole richieste significative filtrate in precedenza.

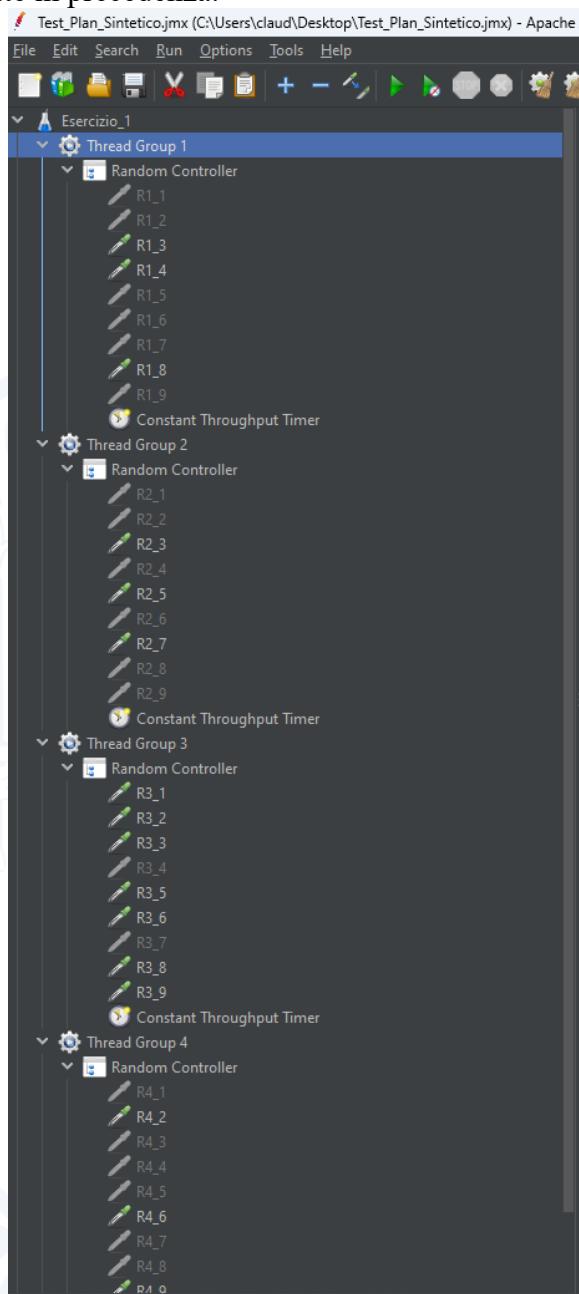


Figura 3.6: Test Plan Workload Sintetico

Queste richieste sono servite per andare a ricreare il workload sintetico dei dati di basso livello (LL').

3.3 Workload Sintetico di Basso Livello LL'

È stato costruito il workload di basso livello LL' come su riportato, dopodiché tramite uno script bash, sono state salvate le colonne relative ai dati di basso livello del server, rendendo i dati pronti per JMP.

```
1 #!/bin/bash
2
3 clear
4
5 # Riavvia il server Apache
6 sudo systemctl restart apache2
7
8 # Attendi 2 secondi
9 sleep 2
10
11 echo server riavviato
12
13 # Genera un nome univoco per il file
14 data=$(date +%Y%m%d-%H%M%S)
15
16 # Avvia vmstat e salva i dati in un file
17 vmstat -n 1 305 > vmstat-$data.csv
18
19 # Elimina la prima riga del file
20 awk 'NR > 1 {print}' vmstat-$data.csv > vmstats-$data.csv
21
22 rm vmstat-$data.csv
23
24 echo report creato
```

Figura 3.7: Script Bash per LL'

3.3.1 Prefiltraggio LL'

Si procede anche in questo caso al prefiltro dei dati, sono state eliminate le colonne: **b** (valori costanti “0”) e **st**(valori costanti “0”), inoltre, tramite la matrice di correlazioni, sono emerse correlazioni tra le colonne: [**buffer,cache**] e [**swpd,free**], si è scelto quindi di eliminare le colonne: **buffer** e **swpd**, in quanto presentano indice di correlazione pari a 1.

3.3.2 PCA Workload Sintetico LL'

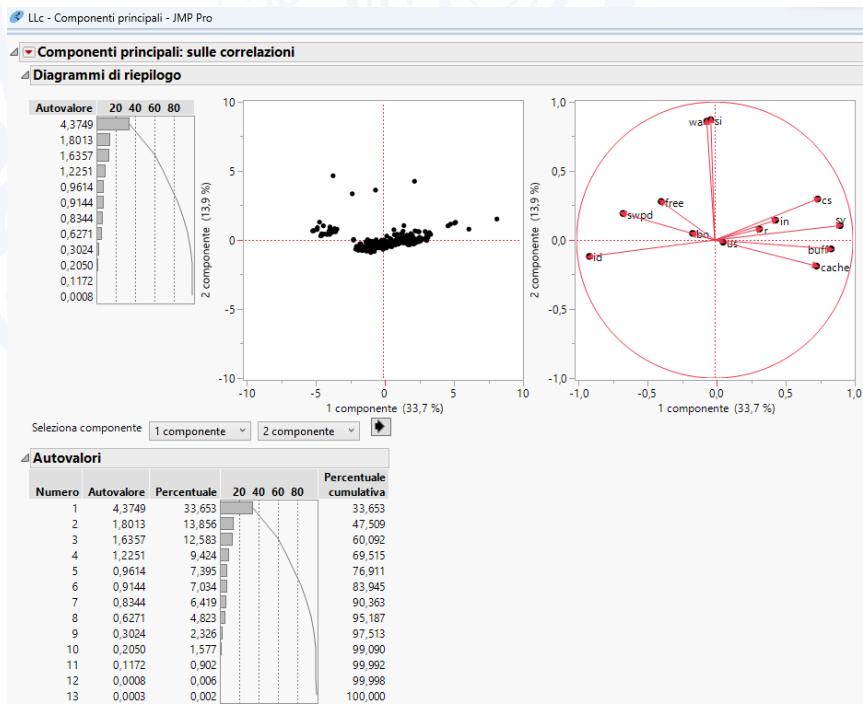


Figura 3.8: PCA LL'

Anche in questo caso sono state scelte 7 componenti che spiegano il 90% della varianza e si è proceduto, come di consueto al clustering.

3.3.3 Clustering

Clusterizzazione gerarchica			
Cronologia di clusterizzazione			
Numero di cluster	Distanza	Leader	Subordinato
55	1,79306132	79	154
52	1,81754075	239	255
51	1,83583931	95	153
49	1,86768899	127	260
48	1,91968612	191	221
47	1,92768464	159	161
46	2,05382921	157	229
45	2,06027944	9	14
44	2,07090483	215	234
43	2,10548576	1	2
42	2,11367061	34	37
41	2,13458421	163	224
40	2,15270343	39	219
39	2,26487097	86	89
38	2,40450158	42	60
37	2,45168054	205	208
36	2,47987735	92	95
35	2,49329650	3	4
34	2,51715937	156	158
33	2,54422585	34	78
32	2,68610718	36	300
31	2,79433318	48	86
30	2,85201999	24	280
29	2,91657676	45	92
28	2,93018062	49	163
27	2,94818142	211	212
26	3,07800514	215	232
25	3,08565554	1	3
24	3,11303275	1	22
23	3,12764583	65	127
22	3,28371034	39	65
21	3,64273997	48	239
20	3,77921375	35	79
19	3,83398759	45	159
18	4,41524466	9	24
17	4,45682374	1	5
16	4,64071382	48	191
15	5,33622517	45	157
14	5,53573363	36	156
13	5,83641740	34	42
12	6,02318783	48	49
11	6,25536333	36	215
10	7,29291721	35	36
9	7,40165489	9	39
8	8,69404186	35	211
7	10,99120698	35	48
6	11,96613948	34	45
5	12,27854707	34	35
4	14,29848737	9	34
3	15,04730628	1	205
2	16,11984220	9	33
1	16,42974434	1	9

Figura 3.9: Clustering LL'

Si scelgono quindi in questo caso 22 cluster, in quanto si nota che dopo 22 cluster non vi sono significative variazioni di distanza.

Per confermare la tesi suddetta, si è scelto di applicare lo script deviance.m per controllare la devianza persa con le scelte effettuate, da cui è scaturito questo risultato:

$$\text{DEV_LOST} = 22\%$$

Quindi si evince che la scelta del cluster è stata corretta, in quanto si è presentato un risultato di devianza persa più che accettabile.

3.3.4 Data Validation – Hypothesis Tests

Nell'ambito dell'analisi statistica multivariata, l'analisi delle componenti principali (PCA) è una tecnica fondamentale per ridurre la dimensionalità dei dati e identificare pattern nascosti. Tuttavia, per interpretare correttamente i risultati della PCA è necessario verificare alcuni presupposti chiave sui dati in ingresso.

Questo capitolo si prefigge di illustrare il processo di verifica di tali presupposti, in particolare la normalità dei dati e l'omoschedasticità delle varianze.

In particolare si andranno a seguire questi passi per ogni componente principale:

- ❖ Verifica della normalità dei dati (LLc e LL'c) -> preferendo il test visivo

- ❖ Se i dati **non** sono normalmente distribuiti, applichiamo un test non parametrico (rank sum test)
- ❖ Oppure verifichiamo che le due distribuzioni abbiano varianza uguale (vartest2)
 - Se hanno stessa varianza applichiamo il two sample t-test
 - Se non hanno stessa varianza applichiamo il sample t-test con varianza diversa

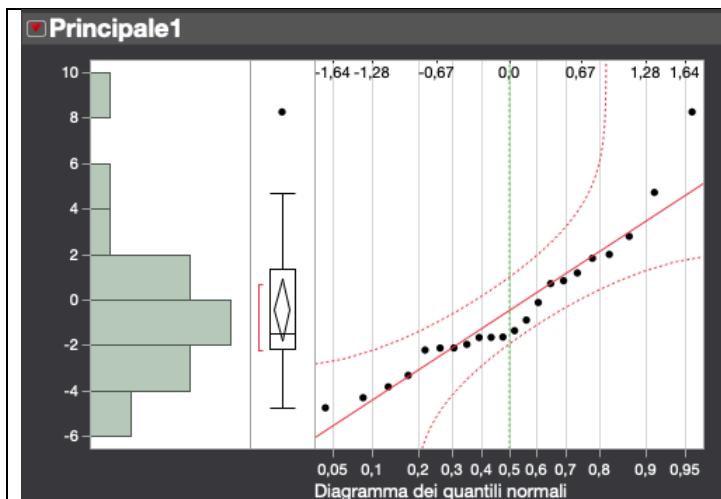


Figura 3.10: Normalità Componente 1 LL

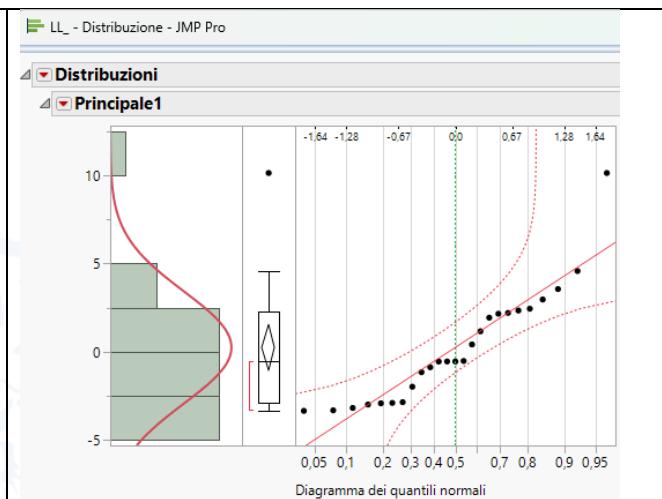
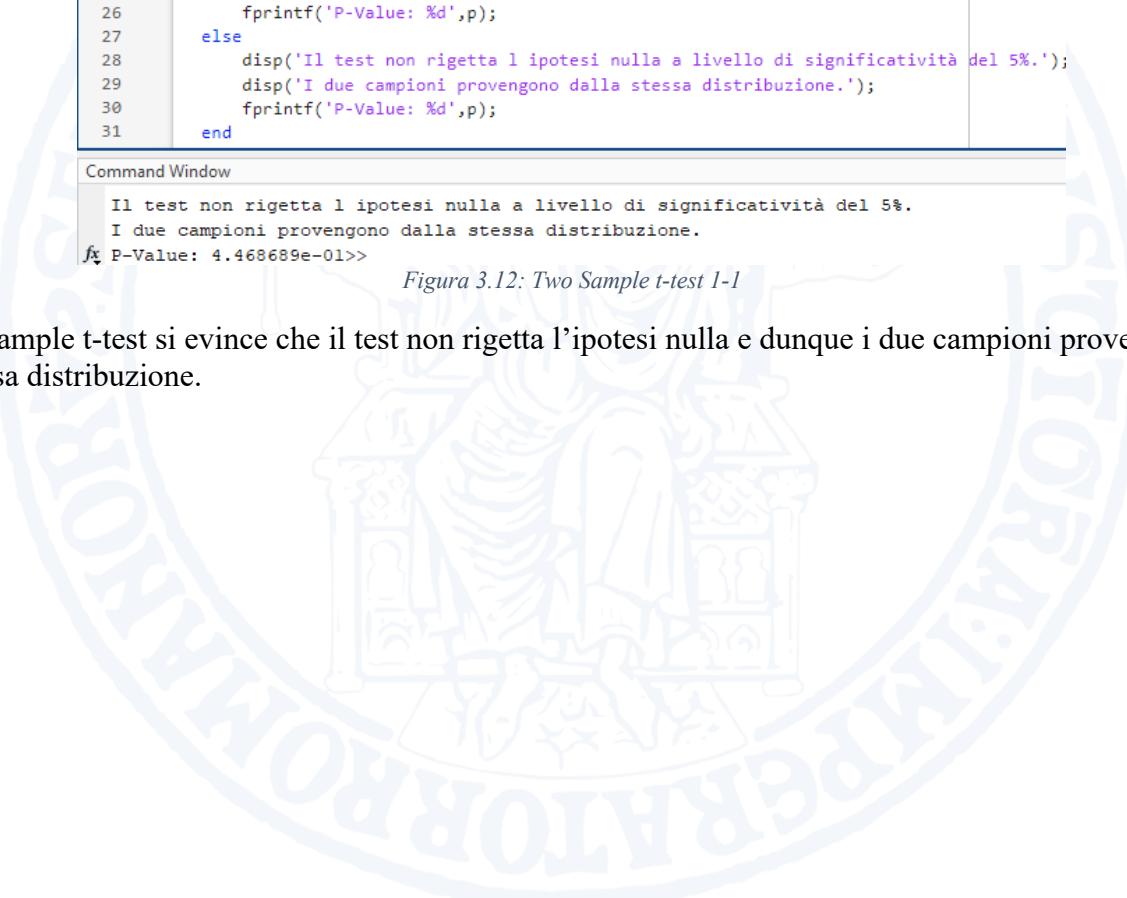


Figura 3.11: Normalità Componente 1 LL'

Dal test visivo si nota che le due distribuzioni sono normali quindi si procede col test della varianza, per capire se le distribuzioni risultano omoschedastiche o meno.

Dallo script matlab **controllo_varianza.m** andiamo a testare la varianza con la funzione **vartest2** che ci ritorna in questo caso un valore $h=0$, quindi l'hyp nulla è respinta ed è possibile asserire che la varianza è uguale, in questo modo possiamo andare ad applicare il two sample t-test.



```

Editor - C:\Users\claud\Desktop\OneDrive - Università di Napoli Federico II\Università\Magistrale\Impianti di Elaborazione\Esercizi
testParametrico.m + [New]

1 clear;
2 clc;
3
4 % Specifica il percorso del file XLSX
5 file_path1 = 'LLc.xlsx';
6 file_path2 = 'LL_Primo_c.xlsx';
7
8
9 % Leggi i dati dal file XLSX 1
10 [data1, text1, raw1] = xlsread(file_path1);
11
12 % Leggi i dati dal file XLSX 2
13 [data2, text2, raw2] = xlsread(file_path2);
14
15 colonna1 = data1(:, 1);
16 colonna2 = data2(:, 1);
17
18 % Esegui il test t di Student
19 [h, p, ci, stats] = ttest2(colonna1, colonna2);
20
21 % Confronta il livello di significatività
22
23 if h == 1
24     disp('Il test rigetta l ipotesi nulla a livello di significatività del 5%.');
25     disp('I due campioni non provengono dalla stessa distribuzione.');
26     fprintf('P-Value: %d',p);
27 else
28     disp('Il test non rigetta l ipotesi nulla a livello di significatività del 5%.');
29     disp('I due campioni provengono dalla stessa distribuzione.');
30     fprintf('P-Value: %d',p);
31 end

```

Command Window

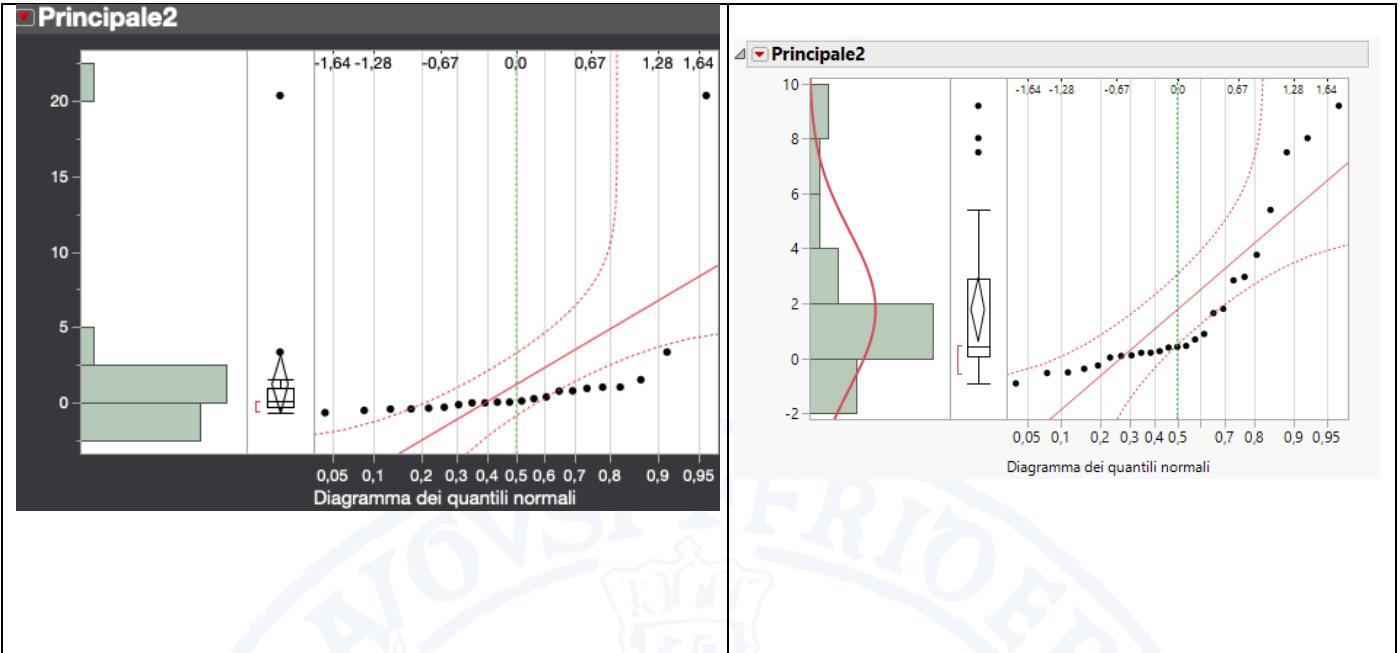
```

Il test non rigetta l ipotesi nulla a livello di significatività del 5%.
I due campioni provengono dalla stessa distribuzione.
fx P-Value: 4.468689e-01>>

```

Figura 3.12: Two Sample t-test I-I

Dal two sample t-test si evince che il test non rigetta l'ipotesi nulla e dunque i due campioni provengono dalla stessa distribuzione.



Dal test visivo si evince come le componenti 2 di LL e LL' non sono normali, per cui si procede con test non parametrico applicando il ranksum test.

```

Editor - C:\Users\claud\Desktop\OneDrive - Università di Napoli Federico II\Università\Magistrale\Impianti di Elaborazione
testParametrico.m Test_Non_Parametrico.m +
```

```

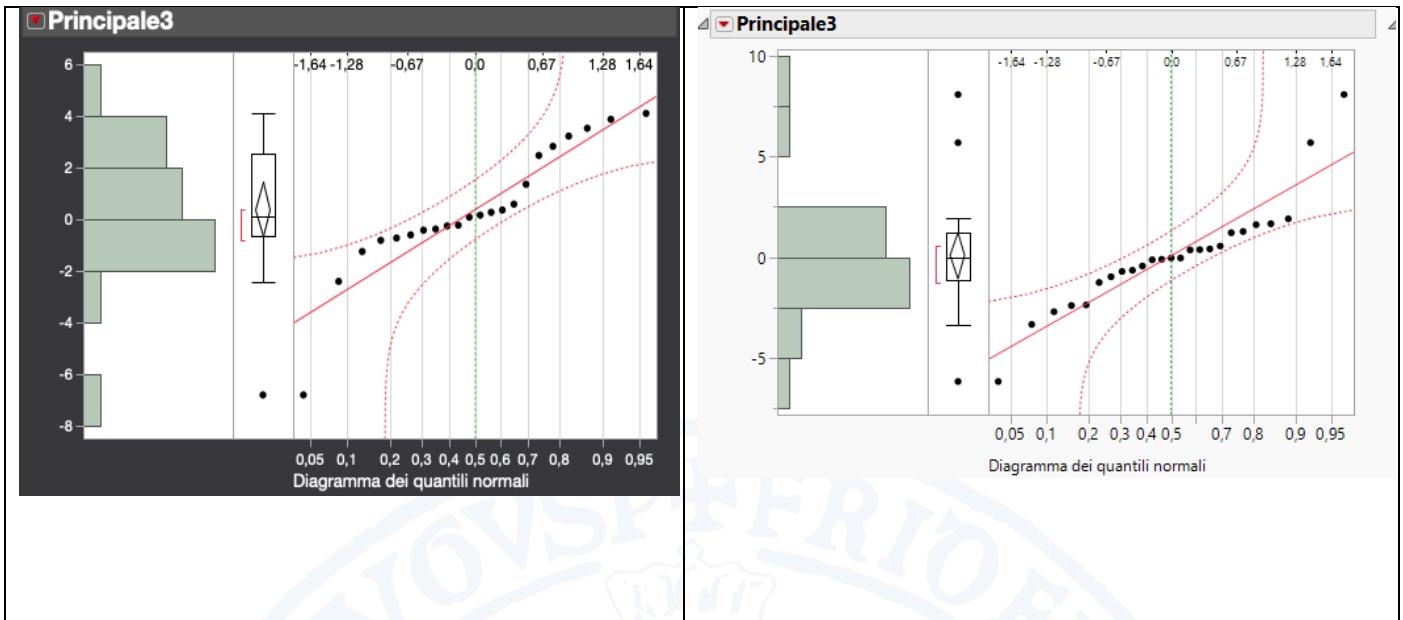
8
9 % Leggi i dati dal file XLSX 1
10 [data1, text1, raw1] = xlsread(file_path1);
11
12 % Leggi i dati dal file XLSX 2
13 [data2, text2, raw2] = xlsread(file_path2);
14
15
16
17 colonna1 = data1(:, 2);
18 colonna2 = data2(:, 2);
19
20 % Esegui il test t di Student
21 [p,h, stats] = ranksum(colonna1, colonna2);
22
23 % Stampa il risultato del test
24 fprintf('Il p-value del test di Wilcoxon è %.4f\n', p);
25
26 % Valuta l'ipotesi nulla
27 if h == 1
28     fprintf('L''ipotesi nulla è rigettata a un livello di significatività del
29 else
30     fprintf('L''ipotesi nulla non può essere rigettata a un livello di signifi
31 end
32
33 % Visualizza le statistiche del test
34 disp('Statistiche del test:');
35 disp(stats);
36
37
```

Command Window

```

Il p-value del test di Wilcoxon è 0.1971
L'ipotesi nulla non può essere rigettata a un livello di significatività del 5%.
Le due distribuzioni non hanno differenze statistiche
Statistiche del test:
    zval: -1.2899
    ranksum: 467
```

Il risultato del ranksum test asserisce che l'hp nulla non può essere rigettata e che quindi non vi sono differenze statistiche tra le due distribuzioni.



Dal test visivo si evince che le due distribuzioni sono normali, per cui si procede con la verifica dell'omoschedasticità.

```

controllo_varianza.m
1 clc;
2 clear;
% Specifica il percorso del file XLSX
4 file_path1 = 'LLc.xlsx';
5 file_path2 = 'LL_Primo_c.xlsx';
6
7
8 % Leggi i dati dal file XLSX 1
9 [data1, text1, raw1] = xlsread(file_path1);
10
11 % Leggi i dati dal file XLSX 2
12 [data2, text2, raw2] = xlsread(file_path2);
13
14 colonna1 = data1(:, 3);
15 colonna2 = data2(:, 3);
16
17
18 % Calcola la varianza delle colonne
19 h = vartest2(colonna1,colonna2);
20
21
22 % Stampa i risultati
23 fprintf('valore di h per il vartest2 -> %.4f\n Le due distribuzioni sono omoschedastiche', h);
24
25

```

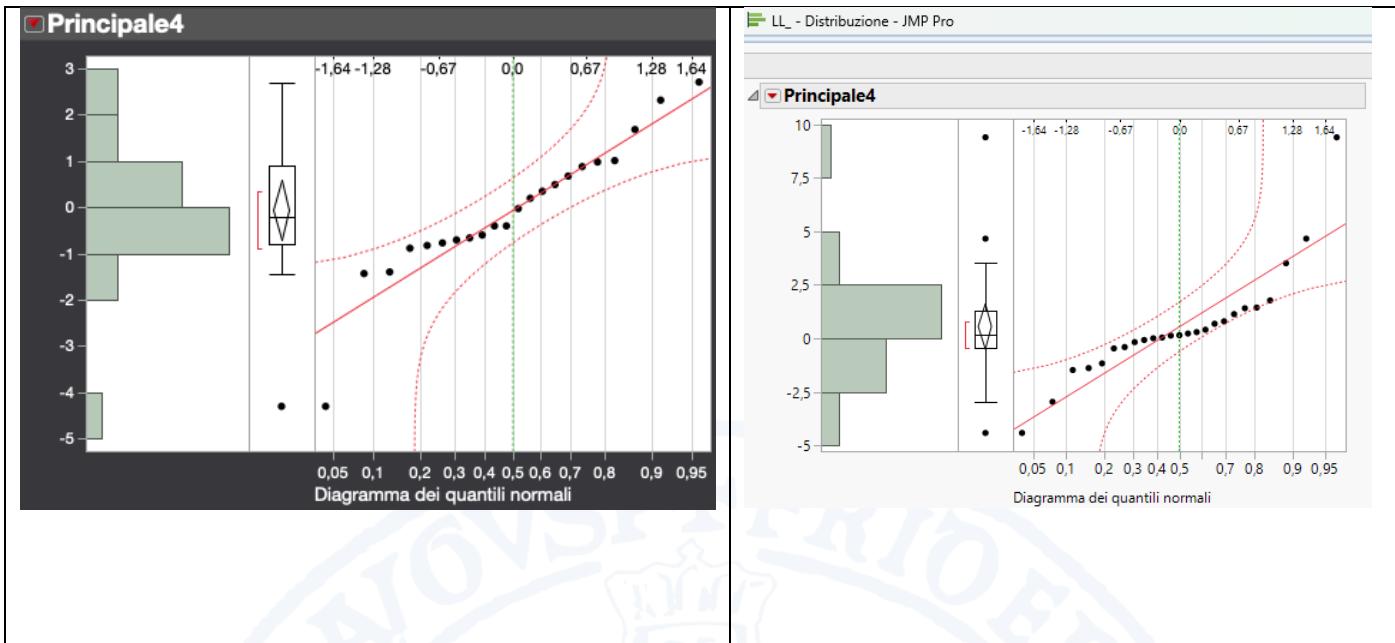
Command Window

```

valore di h per il vartest2 -> 0.0000
fx Le due distribuzioni sono omoschedastiche>

```

Le due distribuzioni risultano omoschedastiche, anche in questo caso il two sample t-test non rigetta l'h_p nulla e dunque si può asserire che anche in questo caso non vi sono differenze statistiche tra i due campioni, che provengono quindi dalla stessa distribuzione.



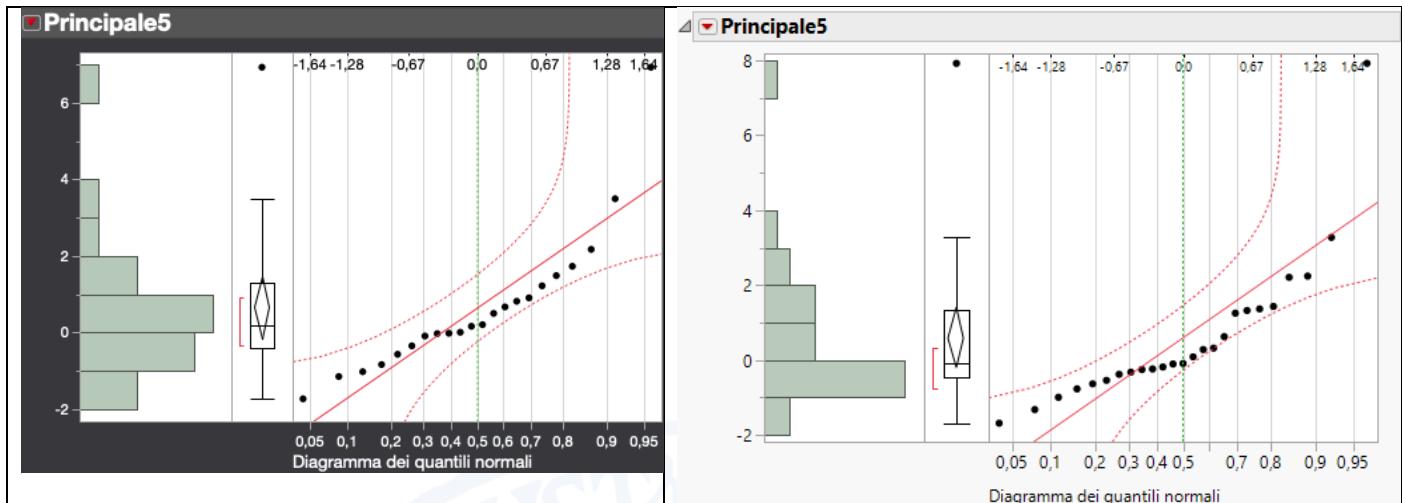
Dal test visivo, in questo caso, è evidente che LL è normale, mentre la componente di LL' no, anche in questo caso è necessario applicare un test non parametrico che ci restituisce questi risultati.

```
17    colonna1 = data1(:, 4);
18    colonna2 = data2(:, 4);
19
20    % Esegui il test t di Student
21    [p,h, stats] = ranksum(colonna1, colonna2);
22
23    % Stampa il risultato del test
24    fprintf('Il p-value del test di Wilcoxon è %.4f\n', p);
25
26    % Valuta l'ipotesi nulla
27    if h == 1
28        fprintf('L''ipotesi nulla è rigettata a un livello di significatività del');
29    else
30        fprintf('L''ipotesi nulla non può essere rigettata a un livello di signifi');
31    end
32
33    % Visualizza le statistiche del test
34    disp('Statistiche del test:');
35    disp(stats);
36
37
```

ommand Window

```
Il p-value del test di Wilcoxon è 0.4240
L'ipotesi nulla non può essere rigettata a un livello di significatività del 5%.
Le due distribuzioni non hanno differenze statistiche
Statistiche del test:
    zval: -0.7995
    ranksum: 460
```

Anche in questo caso otteniamo che i due campioni appartengono alla stessa distribuzione.



Analogamente a quanto detto precedentemente, si andrà a valutare un ranksum test dal momento che una distribuzione è normale (LL) e l'altra no (LL').

```

17 colonna1 = data1(:, 5);
18 colonna2 = data2(:, 5);
19
20 % Esegui il test t di Student
21 [p,h, stats] = ranksum(colonna1, colonna2);
22
23 % Stampa il risultato del test
24 fprintf('Il p-value del test di Wilcoxon è %.4f\n', p);
25
26 % Valuta l'ipotesi nulla
27 if h == 1
28     fprintf('L''ipotesi nulla è rigettata a un livello di significatività del
29 else
30     fprintf('L''ipotesi nulla non può essere rigettata a un livello di signifi
31 end
32
33 % Visualizza le statistiche del test
34 disp('Statistiche del test:');
35 disp(stats);
36
37

```

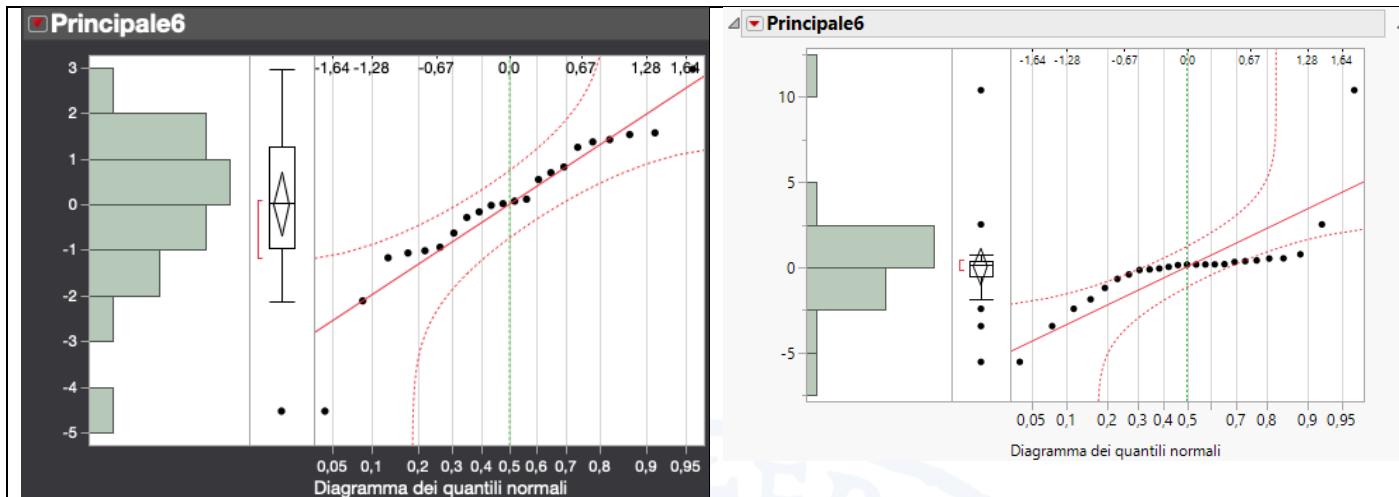
Command Window

```

Il p-value del test di Wilcoxon è 0.7091
L'ipotesi nulla non può essere rigettata a un livello di significatività del 5%.
Le due distribuzioni non hanno differenze statistiche
Statistiche del test:
    zval: 0.3731
    ranksum: 546

```

Anche in questo caso è verificata l'appartenenza alla stessa distribuzione.



Analogamente a quanto detto precedentemente, si andrà a valutare un ranksum test dal momento che una distribuzione è normale (LL) e l'altra no (LL').

```

17     colonna1 = data1(:, 6);
18     colonna2 = data2(:, 6);
19
20     % Esegui il test t di Student
21     [p,h, stats] = ranksum(colonna1, colonna2);
22
23     % Stampa il risultato del test
24     fprintf('Il p-value del test di Wilcoxon è %.4f\n', p);
25
26     % Valuta l'ipotesi nulla
27     if h == 1
28         fprintf('L''ipotesi nulla è rigettata a un livello di significatività del
29     else
30         fprintf('L''ipotesi nulla non può essere rigettata a un livello di signifi
31     end
32
33     % Visualizza le statistiche del test
34     disp('Statistiche del test:');
35     disp(stats);
36
37

```

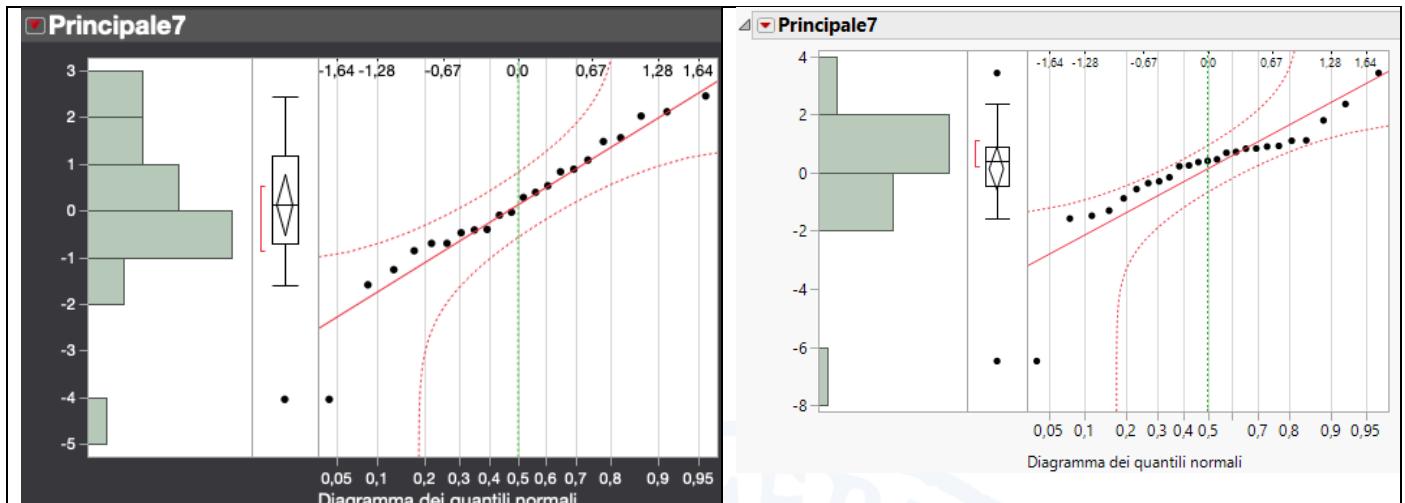
Command Window

```

Il p-value del test di Wilcoxon è 0.7250
L'ipotesi nulla non può essere rigettata a un livello di significatività del 5%.
Le due distribuzioni non hanno differenze statistiche
Statistiche del test:
    zval: 0.3518
    ranksum: 545

```

Anche in questo caso è verificata l'appartenenza alla stessa distribuzione.



Le distribuzioni in questo caso sono normali, quindi si applica il test parametrico che asserisce questi risultati:

```

17 colonna1 = data1(:, 7);
18 colonna2 = data2(:, 7);
19
20 % Esegui il test t di Student
21 [p,h, stats] = ranksum(colonna1, colonna2);
22
23 % Stampa il risultato del test
24 fprintf('Il p-value del test di Wilcoxon è %.4f\n', p);
25
26 % Valuta l'ipotesi nulla
27 if h == 1
28     fprintf('L''ipotesi nulla è rigettata a un livello di significatività del 5%
29 else
30     fprintf('L''ipotesi nulla non può essere rigettata a un livello di significic
31 end
32
33 % Visualizza le statistiche del test
34 disp('Statistiche del test:');
35 disp(stats);
36
37

```

Command Window

```

Il p-value del test di Wilcoxon è 0.7899
L'ipotesi nulla non può essere rigettata a un livello di significatività del 5%.
Le due distribuzioni non hanno differenze statistiche
Statistiche del test:
    zval: -0.2665
    ranksum: 515

```

Da quest' ultimo confronto è possibile affermare che anche queste due componenti appartengono alla stessa distribuzione. È possibile concludere dunque che il workload sintetico non presenta differenze statistiche col workload reale.

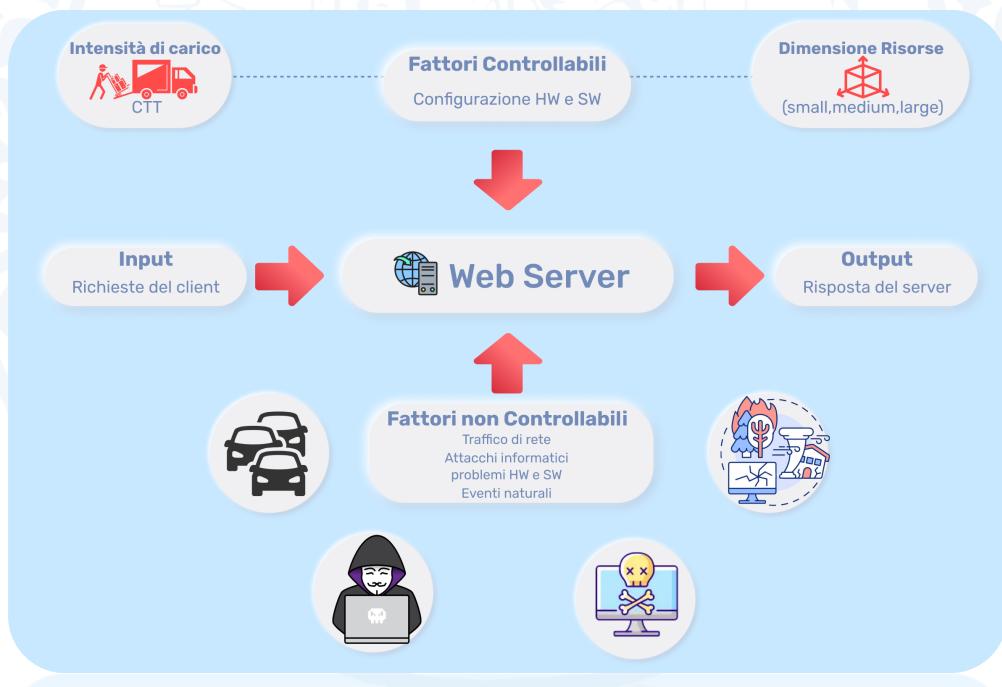
4. Design of Experiment



Il Design of Experiment, noto anche come DOE, rappresenta un metodo statistico utilizzato per identificare i fattori che influenzano una specifica risposta o output del sistema. Tale approccio consente non solo di individuare i fattori in input che incidono sull'output, ma anche di determinare le interazioni tra questi fattori. Attraverso un'analisi accurata degli esperimenti, il DoE permette di separare gli effetti dei vari fattori che potrebbero influenzare le prestazioni del sistema. Questo aiuta a stabilire se un fattore è significativo e/o importante, oppure se le differenze osservate sono dovute a variazioni casuali causate da errori di misurazione o parametri non controllabili. Inoltre, l'utilizzo del Design of Experiment è fondamentale per ridurre il numero di esperimenti richiesti per il testing del sistema, contribuendo così a contenere i costi associati al processo di sperimentazione.

4.1 Configurazione del piano di DOE

La configurazione del Design of Experiment (DOE) si caratterizza come un full factorial design. I fattori che influenzano il sistema sono descritti nel seguente schema:



Ogni fattore presenta diversi livelli:

❖ **Intensità Di carico:**

- Livello 1 – 1500 Richieste al minuto (25% usable)
- Livello 2 – 3000 Richieste al minuto (50% usable)
- Livello 3 – 4500 Richieste al minuto (75% usable)

❖ **Dimensione Risorsa:**

- Livello 1 – Small (17 kB)
- Livello 2 – Medium (100 kB)
- Livello 3 – Large (1,1 MB)

Nome	Dimensione	Modifica
index.html	10,7 kB	20 nov
EXTRASMALL.jpg	16,6 kB	20 ott
MEDIUM.jpg	100,5 kB	14 ott
EXTRALARGE.jpg	1,1 MB	5 ott

$$N = Fattore_1 * Fattore_2 * R_{ripetizioni}$$

Per ogni combinazione di fattori sono state eseguite 5 ripetizioni, per un totale di $3^2 * 5 = 45$ esperimenti condotti tramite JMeter, con ciascun esperimento della durata di un minuto. Pertanto, il tempo totale impiegato per l'esecuzione degli esperimenti è stato di 45 minuti.
La risposta di interesse è il **Response Time**, calcolato come la media dell'Elapsed Time. L'obiettivo principale è determinare l'influenza di tali fattori sulla risposta di interesse. In particolare, si desidera valutare l'**importanza** e la **significatività** dei fattori.

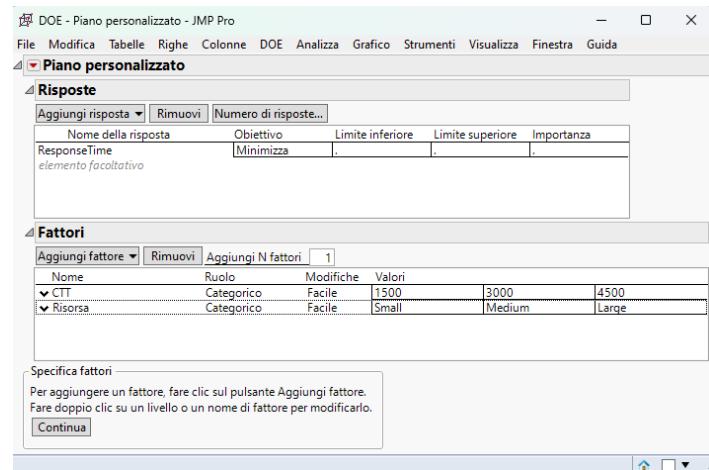
I risultati sono stati organizzati nella seguente disposizione:

1	LL.csv	11 dicembre 2023, 14:17	-- Cartella
1	LM.csv	11 dicembre 2023, 12:57	620 kB Comma...et (.csv)
1	LS.csv	11 dicembre 2023, 12:57	582 kB Comma...et (.csv)
1	ML.csv	11 dicembre 2023, 12:57	610 kB Comma...et (.csv)
1	MM.csv	11 dicembre 2023, 12:57	420 kB Comma...et (.csv)
1	MS.csv	11 dicembre 2023, 12:57	395 kB Comma...et (.csv)
1	SL.csv	11 dicembre 2023, 12:57	414 kB Comma...et (.csv)
1	SM.csv	11 dicembre 2023, 12:57	215 kB Comma...et (.csv)
1	SS.csv	11 dicembre 2023, 12:57	202 kB Comma...et (.csv)
2		11 dicembre 2023, 14:17	-- Cartella
3		11 dicembre 2023, 14:17	-- Cartella
4		11 dicembre 2023, 14:17	-- Cartella
5		11 dicembre 2023, 14:17	-- Cartella

Una cartella per ogni esperimento, ed ogni cartella contiene le varie combinazioni dei livelli discussi precedentemente.

Tramite lo *ScriptMediaElapsed.py* vengono calcolate le medie della colonna elapsed in modo da generare automaticamente il file *mediaElapsed.csv*

Tramite JMP si costruisce il piano del DoE :



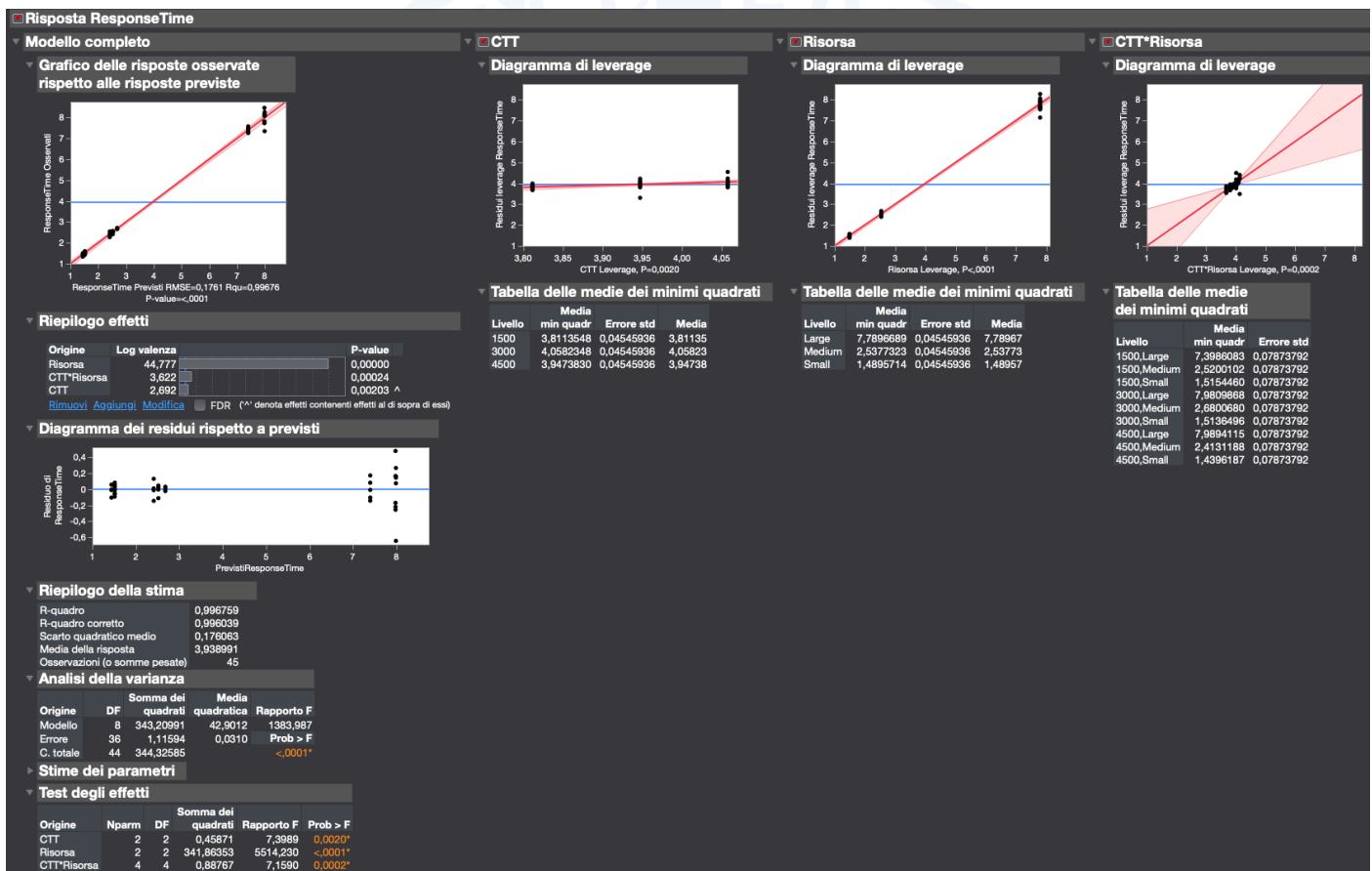
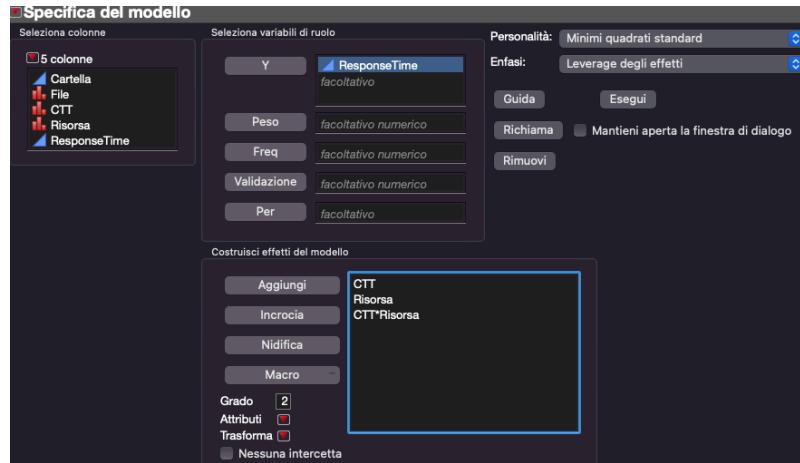
In seguito, si è riempito il piano con i risultati ottenuti dagli esperimenti come mostrato nel seguente estratto:

1	Cartella	File	CTT	Risorsa	ResponseTime
2	1	SL.csv	1500	Large	7.294559099437148
3	4	SL.csv	1500	Large	7.571607254534084
4	3	SL.csv	1500	Large	7.480625
5	2	SL.csv	1500	Large	7.390625
6	5	SL.csv	1500	Large	7.255625
7	1	SM.csv	1500	Medium	2.5159474671669795
8	4	SM.csv	1500	Medium	2.5425
9	3	SM.csv	1500	Medium	2.565978736710444
10	2	SM.csv	1500	Medium	2.408125
11	5	SM.csv	1500	Medium	2.5675
12	1	SS.csv	1500	Small	1.4552845528455285
13	4	SS.csv	1500	Small	1.546875
14	3	SS.csv	1500	Small	1.5146966854283928
15	2	SS.csv	1500	Small	1.5978736710444028
16	5	SS.csv	1500	Small	1.4625
17	1	ML.csv	3000	Large	7.760890609874153
18	4	ML.csv	3000	Large	8.461290322580645
19	3	ML.csv	3000	Large	7.724104549854792
20	2	ML.csv	3000	Large	8.147741935483872
21	5	ML.csv	3000	Large	7.81090674411003
22	1	MM.csv	3000	Medium	2.685483870967742
23	4	MM.csv	3000	Medium	2.7115198451113263
24	3	MM.csv	3000	Medium	2.6616129032258065
25	2	MM.csv	3000	Medium	2.6569861245563087
26	5	MM.csv	3000	Medium	2.6847370119393354
27	1	MS.csv	3000	Small	1.521135850274282
28	4	MS.csv	3000	Small	1.420780897063569
29	3	MS.csv	3000	Small	1.5692158760890609
30	2	MS.csv	3000	Small	1.500161342368506

4.2 Analisi dell'importanza

Per valutare l'importanza di un fattore all'interno del piano, è necessario calcolare alcuni parametri:

- ❖ **SS_{ctt}**(Sum of Square factor ctt): Questo parametro misura la variazione spiegata dall'intensità del fattore.
- ❖ **SS_{risorsa}**(Sum of Square factor risorsa): Questo parametro misura la variazione spiegata dalla dimensione della risorsa.
- ❖ **SS_{CttRisorsa}**(Sum of Square factor ctt*risorsa): Questo parametro misura la variazione spiegata dall'interazione tra la dimensione della risorsa e l'intensità.
- ❖ **SS_{Errore}**: Questo parametro misura la variazione spiegata dall'errore.



Tramite il tool JMP è possibile valutare le variazioni tramite le Sum of Square (SS) dei fattori, delle interazioni dei fattori, dell'errore e quella totale per valutare la percentuale di variazione spiegata da ognuno.

Pertanto:

$$\frac{SS_{ctt}}{SS_{totale}} = 0,0013321 \approx 0,13\%$$

$$\frac{SS_{risorsa}}{SS_{totale}} = 0,9928488 \approx 99,2\%$$

$$\frac{SS_{ctt*risorsa}}{SS_{totale}} = 0,0025779 \approx 0,26\%$$

$$\frac{SS_{errore}}{SS_{totale}} = 0,0032409 \approx 0,32\%$$

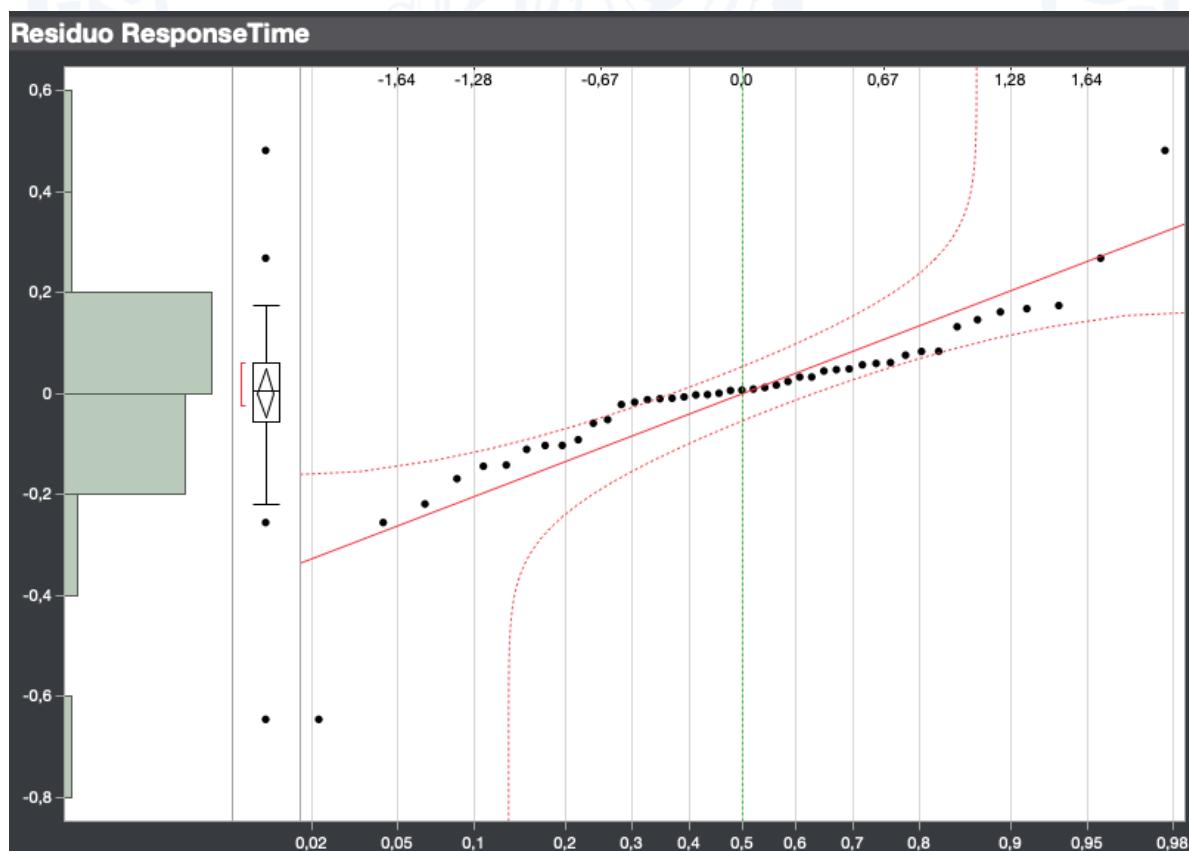
Dalla valutazione emerge che il **fattore con l'importanza maggiore** è la **Dimensione della Risorsa**, il quale **spiega il 99,2% della varianza totale** (complessiva). Il fattore del numero di richieste al minuto (ctt) spiega solamente il 0,13% della varianza totale, mentre il fattore d'interazione tra la Dimensione della Risorsa e ctt spiega circa lo 0,23%. I termini di errore spiegano all'incirca lo 0,32% della varianza complessiva, indicando che solo il 0,23% della varianza totale nei dati è attribuibile a un errore sperimentale di misura.

4.3 Analisi della significatività statistica dei fattori

Si valuta quindi la percentuale di variabilità spiegata da un fattore rispetto a quella spiegata dall'errore. Per studiare la **significatività** dei fattori considerati, si applica un metodo chiamato **ANOVA** (Analysis Of Variance). Il metodo specifico da utilizzare dipende da alcune caratteristiche della distribuzione, in particolare la normalità e l'omoschedasticità.

4.3.1 Analisi della normalità

Si procede con la verifica dell'approssimazione degli errori a una distribuzione normale. Questo processo include il calcolo dei residui e l'analisi del QQPlot dei residui per valutarne visivamente la normalità.

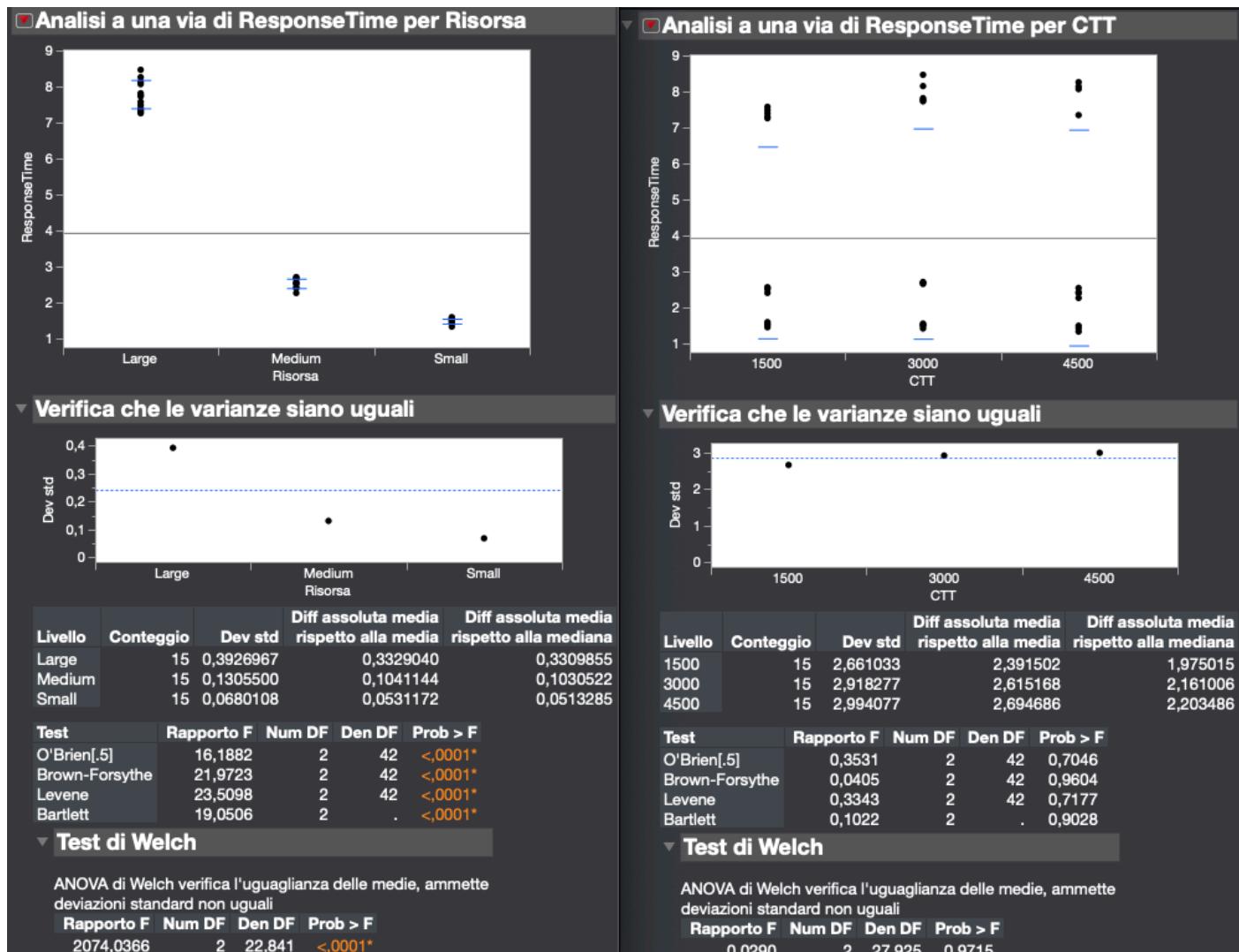


Da un test visivo la distribuzione appare non normale, come ulteriore verifica viene applicato anche il test di Shapiro-Wilk.

Test della bontà di adattamento	
W	Prob<W
Shapiro-Wilk 0,8659166	<.0001*
A ²	P-value simulato
Anderson-Darling 1,7375538	0,0004*
Nota: Ho = i dati provengono dalla distribuzione Normale. I p-value bassi rifiutano Ho.	

Anche il test conferma le conclusioni del test visivo rifiutando H_0 .

4.3.2 Analisi omoschedasticità



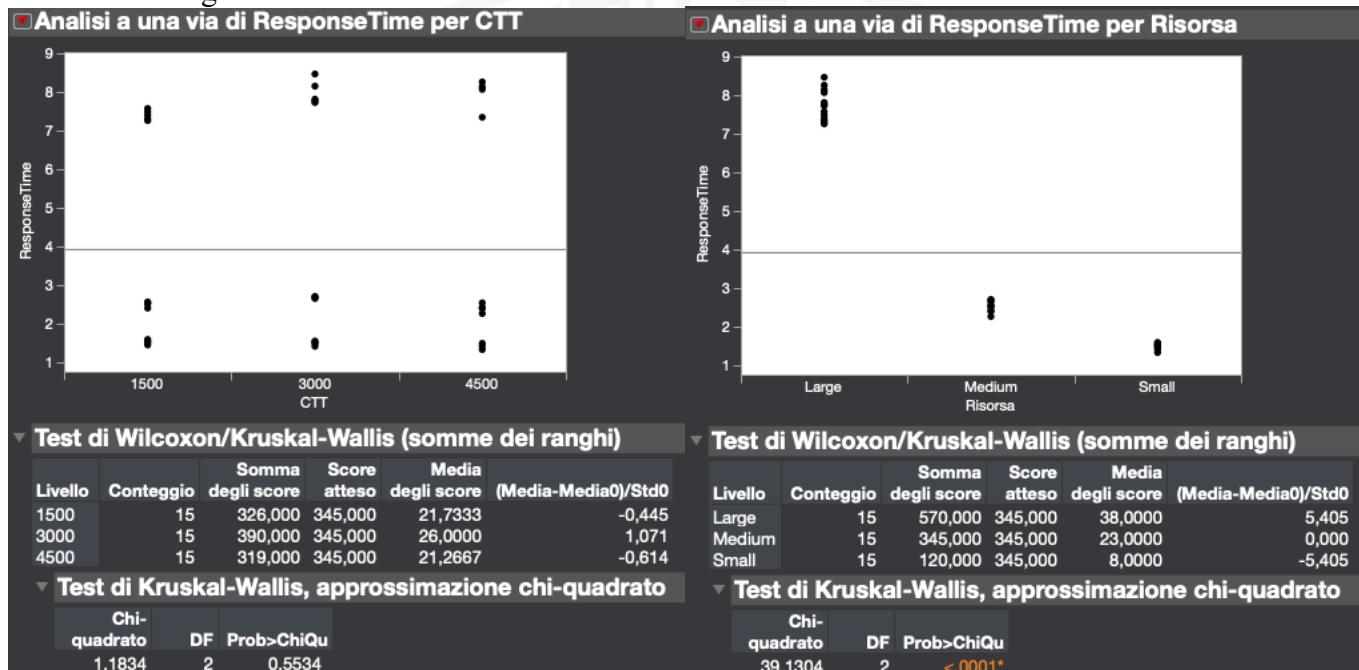
Dal test visivo risulta eteroschedastico il fattore Risorsa mentre il fattore CTT risulta omoschedastico.

4.3.3 Analisi significatività

Dalle analisi della normalità e dell'omoschedasticità, è possibile valutare quale test applicare alla seguente tabella.

Normalità	Omoschedasticità	Anova Test
Si	Si	F-test
No	Si	Kruskal-Walls test
Si	No	Welch's test
No	No	Kruskal-Walls test o Friedman test

Per valutare la significatività dei fattori andiamo ad utilizzare



Utilizzando Kruskal-Walls test per entrambi i fattori (per CTT si sarebbe potuto utilizzare anche Friedman test), si nota che l'unico fattore significativo è la dimensione della **risorsa**.

Il p-value associato alla Dimensione della Risorsa è significativamente basso, il che porta al rigetto dell'ipotesi di non significatività statistica. D'altra parte, l'Intensità delle risorse non risulta statisticamente significativa poiché il p-value indica che l'ipotesi nulla (H_0) non può essere rigettata.

Nota: Poiché i dati non seguono una distribuzione normale, il test di Kruskal-Wallis utilizza la distribuzione Chi-square per valutare il p-value.

In conclusione, si può affermare che la dimensione della risorsa è un fattore sia importante che significativo.

5. Benchmark

Un benchmark è un test standardizzato progettato per valutare le prestazioni di un sistema, un componente hardware o software, o un'applicazione. L'obiettivo di un benchmark è misurare le capacità di un sistema in modo ripetibile e confrontabile. I risultati del benchmark forniscono informazioni sulla velocità, l'efficienza e la stabilità del sistema o del componente testato.

Per realizzare tale test verrà utilizzato il programma Nbody. Esso simula l'evoluzione di N corpi celesti, sotto l'influenza della forza di gravità.

Lo script launch_nbody.sh consente di avviare il benchmark al variare di N e del numero di ripetizioni impostate. Contestualmente ne raccoglie i tempi di esecuzione in millisecondi e li fornisce in output.

I sistemi che andremo a confrontare sono i seguenti:



Modello	Inspiron 15 Plus
Processore	Intel® Core™ i7-11800H (4,6 GHz)
Ram	16 GB DDR4 a 3.200 MHz
Scheda Grafica	NVIDIA(R) GeForce RTX (TM) 3050 Ti (GDDR6 4GB)
Storage	SSD PCIe NVMe M.2 da 1 TB
Sistema Operativo	Windows 11 Pro 23H2

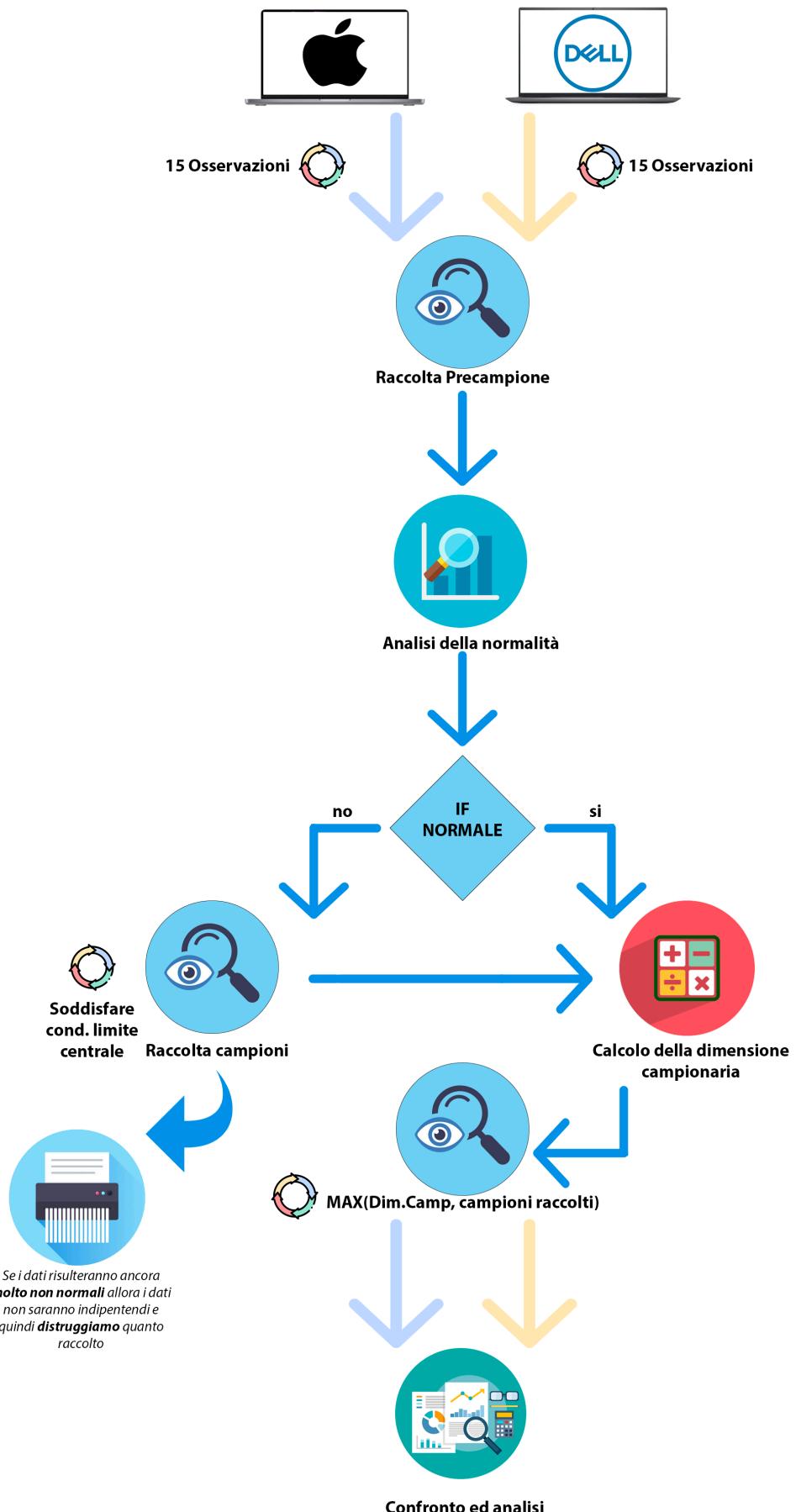
Macchina DELL (Claudio)



Modello	MacBook Pro (15-inch, 2016)
Processore	Intel Core i7 quad-core 2,6 GHz
Ram	16 GB 2133 MHz LPDDR3
Scheda Grafica	Radeon Pro 450 2 GB
Storage	SSD PCIe NVMe M.2 da 256 GB
Sistema Operativo	macOS Monterey 12.7.1

Macchina APPLE (Mauro)

Tale analisi sarà articolata in diverse fasi:



5.1 Raccolta pre-campione

La fase preliminare di questa analisi implica la raccolta dei dati attraverso operazioni di misurazione diretta. La metrica adottata è il tempo di esecuzione della simulazione. Una condizione primaria essenziale da soddisfare è l'indipendenza tra i campioni raccolti.

Al fine di garantire l'indipendenza tra i campioni raccolti durante ciascuna misurazione, le macchine soggette all'analisi sono state riavviate. Questa procedura è stata adottata al fine di attenuare gli effetti derivanti dall'ottimizzazione delle prestazioni a causa dell'utilizzo della cache.

Il comando adoperato per l'esecuzione della simulazione e l'acquisizione dei dati è il seguente:

```
1. ./launch_nbbody.sh -r 5 -n 1000000 >> 1000000.txt
```

Nella suddetta istruzione, il valore "5" indica il numero di ripetizioni della simulazione, mentre "1000000" rappresenta il numero di corpi oggetto di simulazione. L'output prodotto dal comando viene direzionato verso un file di testo al fine di consentirne un'analisi successiva.

Tale comando sarà eseguito quindici volte su ciascuna macchina, con un riavvio di quest'ultima ad ogni iterazione, e con un intervallo di attesa adeguato per consentire un'adeguata stabilizzazione al momento dell'accensione delle macchine.

Questo processo è stato ripetuto utilizzando differenti numeri di corpi al fine di rendere più dettagliata l'analisi. Sono state quindi simulate configurazioni con 1.000, 100.000 e 1.000.000 di corpi.

I file generati saranno composti ciascuno da 75 righe, rappresentanti le 5 ripetizioni per i 15 esperimenti condotti.

Attraverso l'utilizzo dello script Python denominato "scriptMedia.py", mediante l'inserimento del nome della cartella contenente i dati, verrà calcolata la media delle ripetizioni per ciascun file, ottenendo quindi un file CSV composto da 15 righe.

Di seguito vengono presentate le medie delle misure ottenute:

1000.txt	100000.txt	1000000.txt
468,6	38227,4	368014,4
407,2	37377,8	379654,4
386,2	36933,2	387272,8
521,4	39267,2	356732,0
507,0	42764,8	372601,8
411,8	38060,6	376290,0
418,4	37231,8	381207,8
362,4	37664,8	373603,8
394,8	36478,6	447085,6
411,0	35572,2	444004,4
467,6	37741,4	405063,4
412,8	37040,2	368641,4
537,2	38412,4	436539,4
447,8	44190,0	379262,8
455,2	36999,2	384035,2

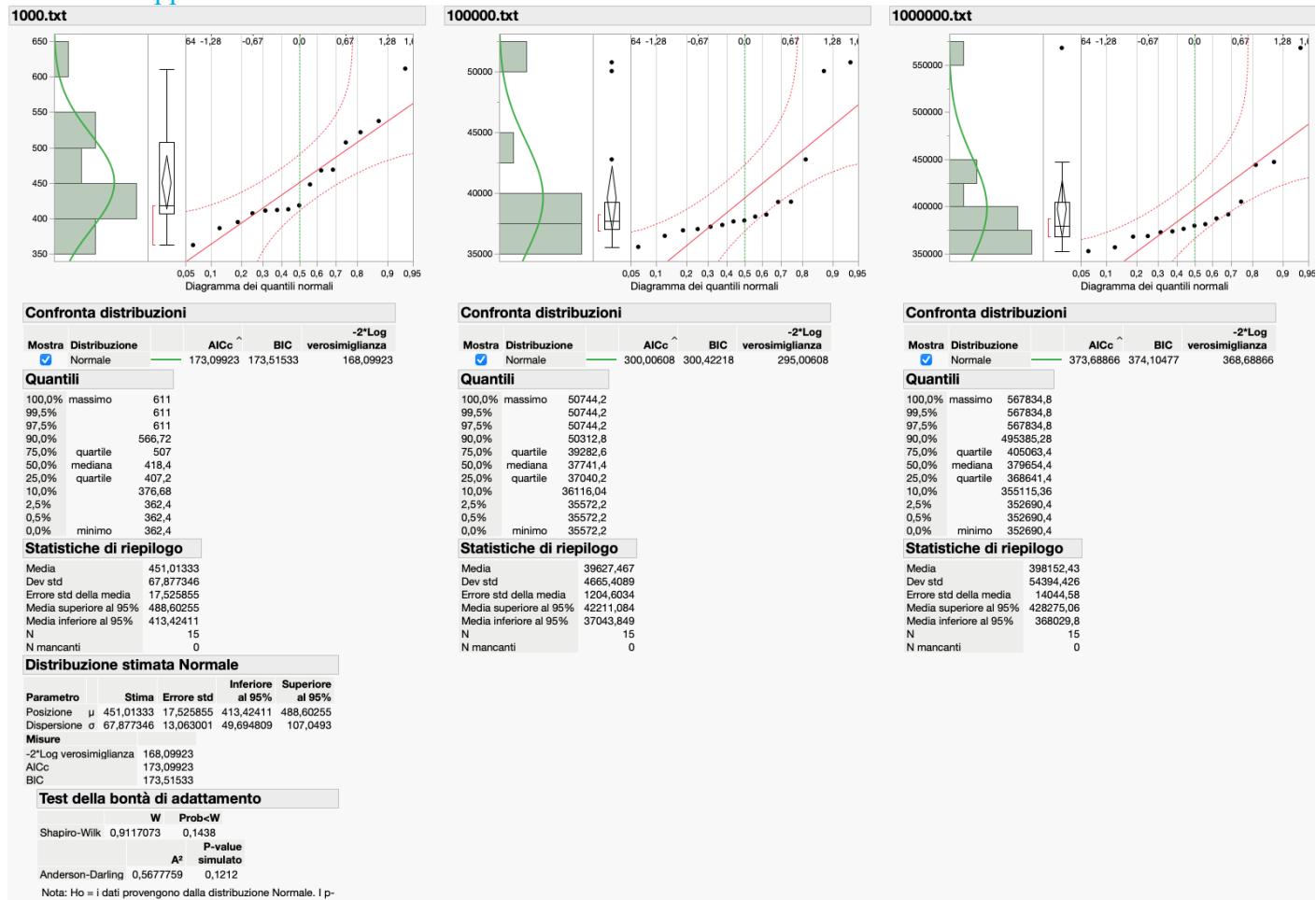
Dati Apple Mauro

Medie 1K	Medie 100K	Medie 1M
172,6	15678,6	153772,8
180,8	16195,8	154693,6
181	16612,8	159744,6
182	15263	166718
170,4	15565,6	163593,8
169,4	15069,2	169366
169,6	15451	162503,6
169,2	16218,2	150754,6
189,8	15442,4	175678,4
173,4	16522,4	177308,4
173,2	15908,8	175961,6
177,2	16153	152951,4
170,4	15456,8	157465,2
176,6	16308,6	156830
166,4	15342,8	161204,2

Dati Dell Claudio

5.2 Analisi della normalità delle distribuzioni

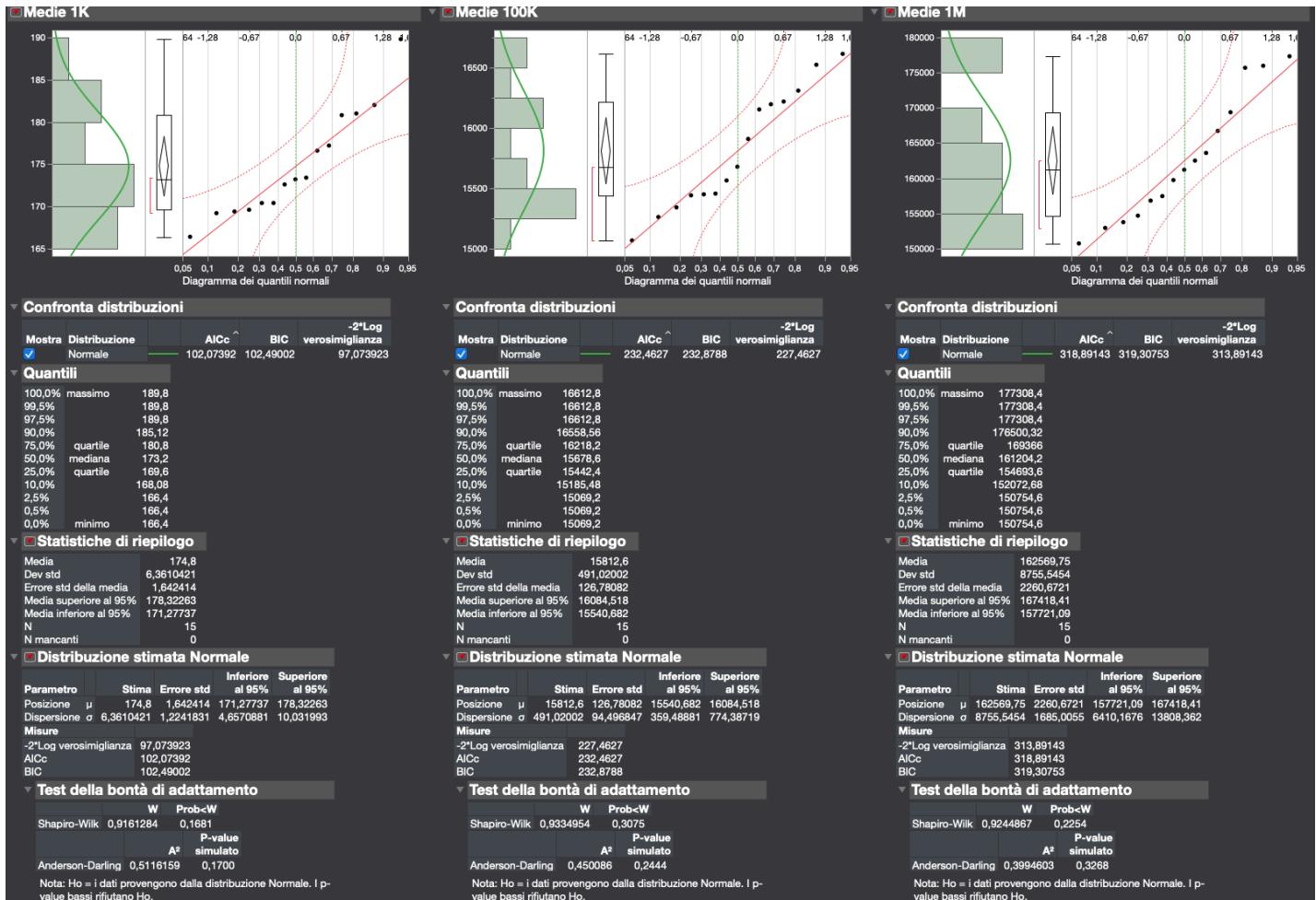
Macchina Apple



Dall'analisi visiva condotta su JMP risulta solo la distribuzione dei mille corpi normale, confermato anche dal test Shapiro-Wilk.

Andremo quindi a raccogliere ulteriori 30 campioni per centomila corpi ed un milione di corpi.

Macchina Dell



Da questi grafici si può evincere che tutte le distribuzioni risultano fortemente normali e quindi possiamo procedere al calcolo della dimensione campionaria.

5.3 Dimensione Campionaria

I pre-campioni sono risultati necessari per il calcolo della dimensione campionaria n, ovvero la dimensione effettiva di ogni campione che garantisce un prefissato errore E sulle medie.

Tale dimensione sarà calcolata come:

$$n = \left(\frac{t_{\alpha/2, n-1} * \sigma}{E} \right)^2$$

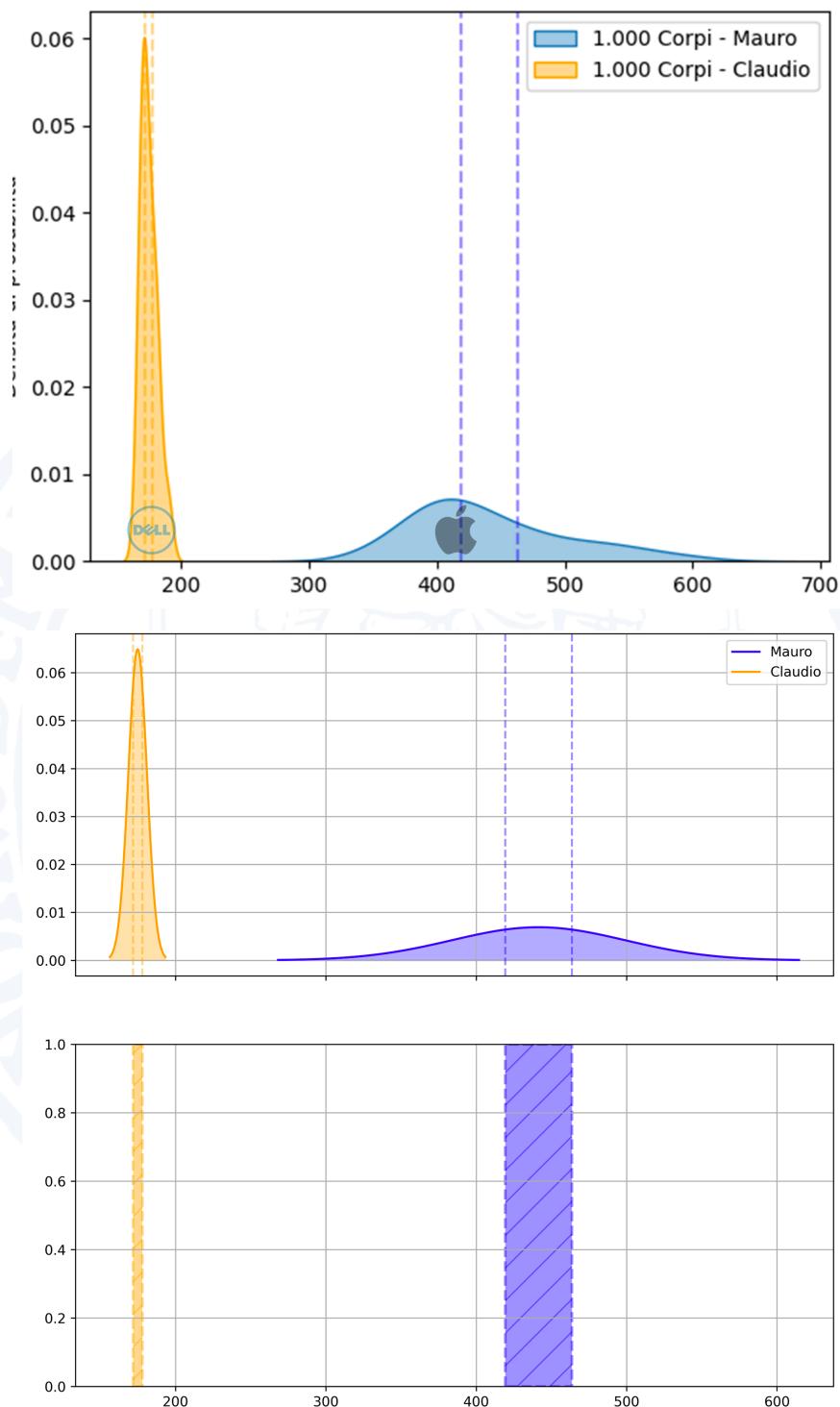
Tramite lo script Matlab dimensioneCampionaria.m abbiamo calcolato in maniera automatica la dimensione campionaria per ogni configurazione di simulazione per entrambe le macchine.

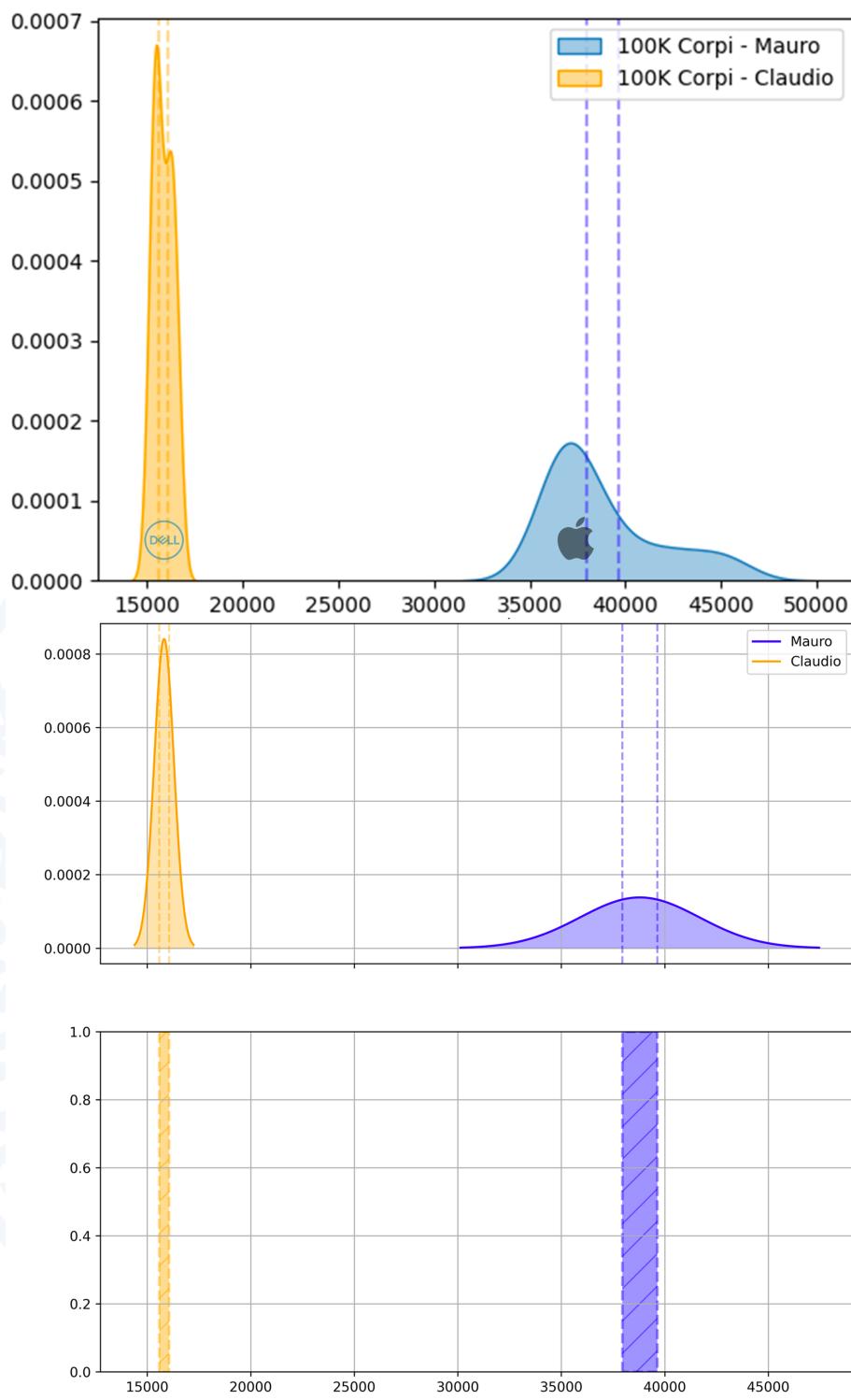
Dell			Apple		
1K corpi	100K corpi	1M corpi	1K corpi	100K corpi	1M corpi
2.4366	1.7742	5.3372	25.2545	9.2091	15.6640

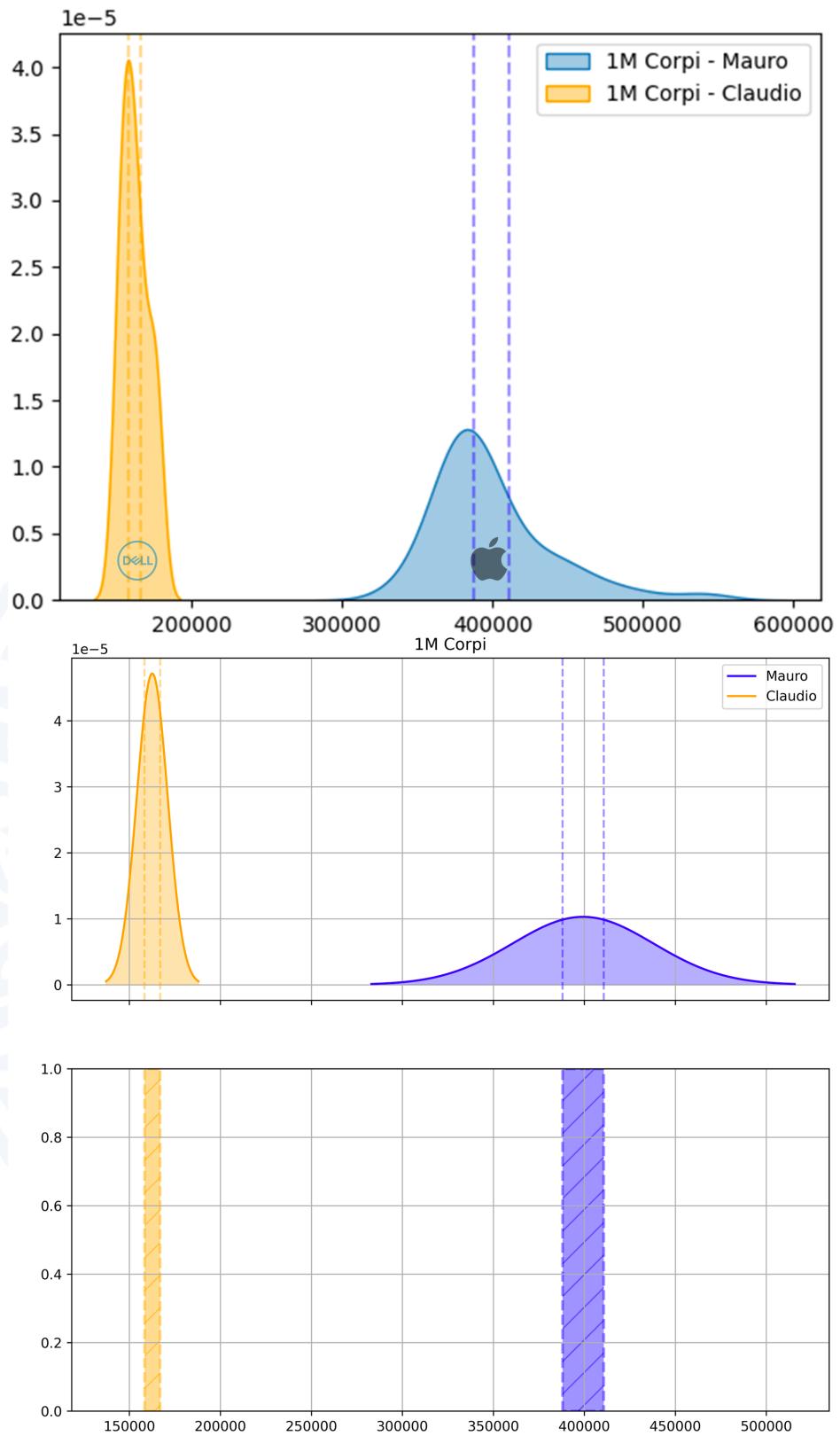
Confrontando il numero di esperimenti effettuati rispetto alle dimensioni campionarie calcolate, effettuiamo ulteriori 11 esperimenti per la configurazione a 1K corpi per la macchina Apple.

5.4 Differenza statistica

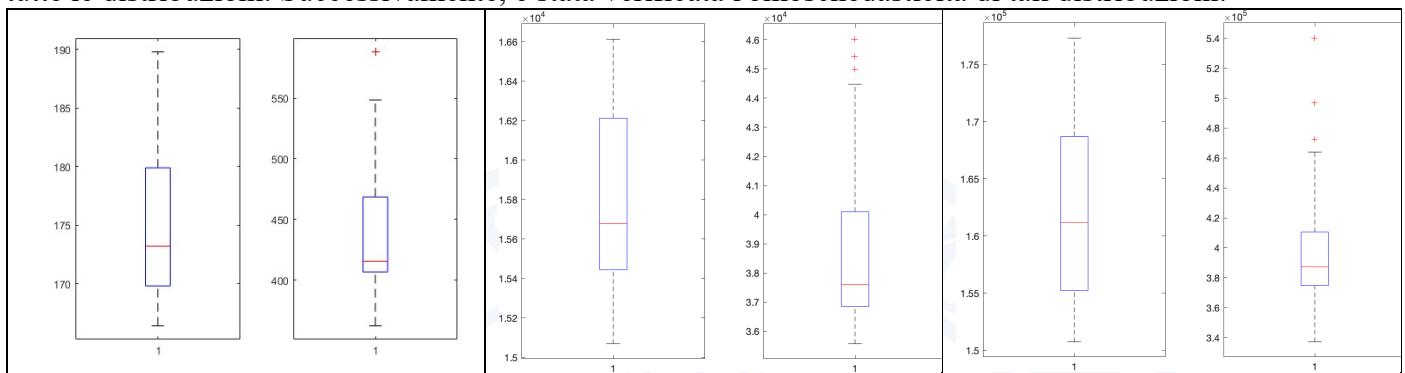
Andiamo quindi a valutare per ogni configurazione se ci sono differenze statistiche tra le due macchine. In primo luogo, andiamo ad effettuare una analisi visiva con gli overlapping degli intervalli di confidenza.







Come si può desumere dall'analisi dei grafici, in nessuna delle configurazioni è stata osservata una sovrapposizione degli intervalli di confidenza. Al fine di confermare ulteriormente i risultati ottenuti, è stato deciso di condurre un test di ipotesi sulla differenza delle medie dei due campioni, al fine di determinare se tale differenza sia statisticamente significativa. In altre parole, si intende stabilire se i due campioni provengano dalla stessa popolazione d'origine. Dopo aver verificato la normalità di alcune configurazioni dei pre-campioni e le condizioni del teorema del limite centrale per le restanti, è stata confermata la normalità di tutte le distribuzioni. Successivamente, è stata verificata l'omoschedasticità di tali distribuzioni.



Le distribuzioni risultano eteroschedastiche.

Applichiamo quindi un *Two-sample t-test (unpooled)*.

```
Editor - C:\Users\claud\Desktop\OneDrive - Università di Napoli Federico II\Università\Magistrale\Impianti di Elaborazione\Esercizi\Benchmark\Confronto\BoxPlot...
BoxPlot_Cannel.m + [x]
28 [h, p, ci, stats] = ttest2(data_claudio1, data_mauro1, 'Vartype', 'unequal');
29
30 % Visualizza i risultati
31 disp('Risultati del two-sample t-test unpooled Claudio1-Mauro1:');
32 disp(['Statistiche t = ', num2str(stats.tstat)]);
33 disp(['Valore p = ', num2str(p)]);
34 disp(['Intervallo di confidenza = [', num2str(ci(1)), ', ', num2str(ci(2)), ']']);
35
36 % Verifica l'ipotesi nulla
37 if h
38     disp('L ipotesi nulla è rifiutata.');
39 else
40     disp('L ipotesi nulla non è rifiutata.');
41 end
42
43 disp('');
44
45 [h, p, ci, stats] = ttest2(data_claudio2, data_mauro2, 'Vartype', 'unequal');
46
```

Command Window

```
Risultati del two-sample t-test unpooled Claudio1-Mauro1:
Statistiche t = -22.8506
Valore p = 9.7281e-19
Intervallo di confidenza = [-290.5819, -242.6181]
L ipotesi nulla è rifiutata.

Risultati del two-sample t-test unpooled Claudio2-Mauro2:
Statistiche t = -50.6752
Valore p = 4.6496e-45
Intervallo di confidenza = [-23889.1012, -22068.1077]
L ipotesi nulla è rifiutata.

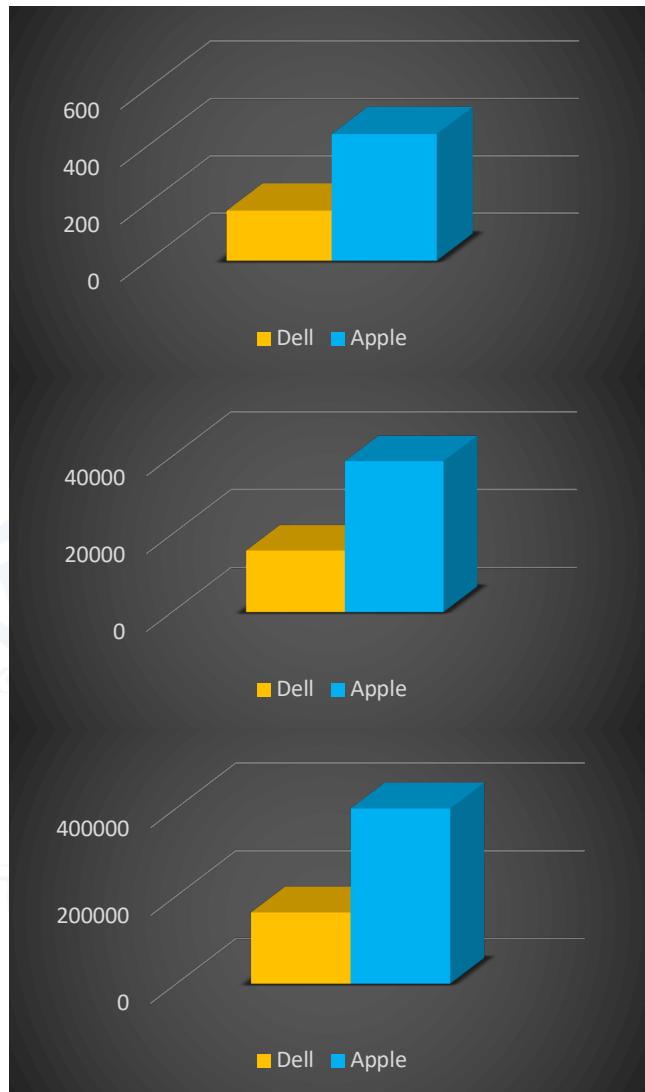
Risultati del two-sample t-test unpooled Claudio3-Mauro3:
Statistiche t = -50.6752
Valore p = 4.6496e-45
Intervallo di confidenza = [-23889.1012, -22068.1077]
L ipotesi nulla è rifiutata.
f2 >>
```

Dal test condotto viene confermato che i dati in esame sono statisticamente differenti.

5.5 Risultati e conclusioni

Dalle analisi effettuate emerge una significativa disparità nelle prestazioni tra le due macchine. La macchina Dell, caratterizzata da un hardware più recente di almeno cinque anni rispetto alla macchina Apple oggetto di confronto, manifesta una superiorità in termini di performance in tutte le configurazioni di corpi simulati considerate.

Di seguito sono mostrate le medie dei tempi di esecuzione per **1k, 100k ed 1M** di corpi.



*Valori più bassi della media dei tempi di esecuzione rappresentano migliori prestazioni della macchina.

6. Regressione

Un modello regressivo è un tipo di modello statistico che mira a identificare una relazione tra una variabile dipendente (risposta o stima) e una o più variabili indipendenti (predittive). La relazione è formulata attraverso una funzione lineare che descrive il modello. L'obiettivo consiste nell'esaminare il trend, ossia nell'analizzare una serie temporale al fine di individuare una tendenza sottostante. Quest'ultima può manifestarsi in modo positivo, negativo o essere nulla e viene utilizzata per anticipare i valori futuri della serie temporale.

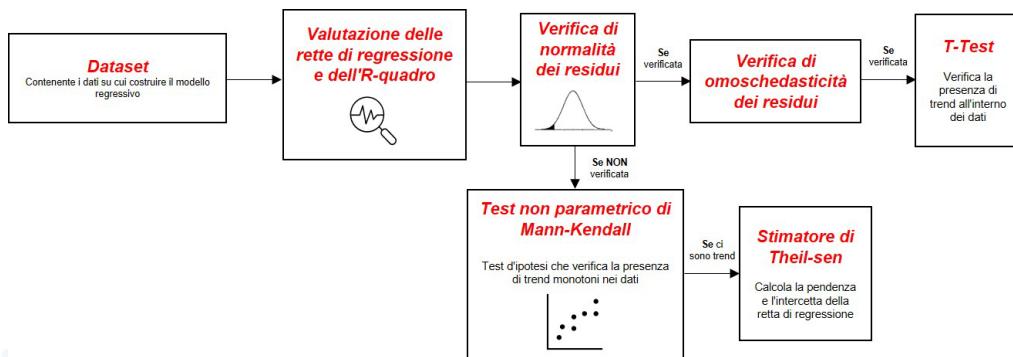


Figura 6.1: Schema Modello Regressivo

6.2 Dataset Mail Server 1

Il dataset è così formato:

- ❖ Variabili di risposta:
 - Nbyte
 - Byte_rec
 - Byte_sent
- ❖ Variabili di predizione
 - Observation

6.2.1 Valutazione delle rette di regressione e R²

È applicato un modello regressivo lineare, il cui risultato è una funzione lineare del predittore. La stima lineare delle variabili è valutata su JMP, come illustrato nella Figura 6.2: Stima lineare e valutazione R². Nella stima, viene fornito l'R-quadrato, ossia il coefficiente di determinazione, che valuta la relazione tra la variabilità dei dati e l'accuratezza del modello statistico utilizzato. Questo indicatore misura quanto bene le previsioni del modello si adattano ai dati effettivi. L'R-quadrato assume valori compresi tra 0 e 1, dove 1 indica una spiegazione perfetta della varianza dei dati da parte del modello, mentre 0 indica una mancanza totale di spiegazione della varianza dei dati. In tutti e tre i casi, l'R-quadrato mostra un valore molto basso, indicando che il modello regressivo lineare utilizzato presenta una scarsa adattabilità ai dati, e quindi le prestazioni predittive sono insoddisfacenti.

Nel caso si voglia impiegare il modello regressivo per il rilevamento delle tendenze, è possibile che il coefficiente R-quadrato risulti basso. Tuttavia, è essenziale verificare la significatività dei parametri della retta regressiva attraverso un opportuno test statistico.

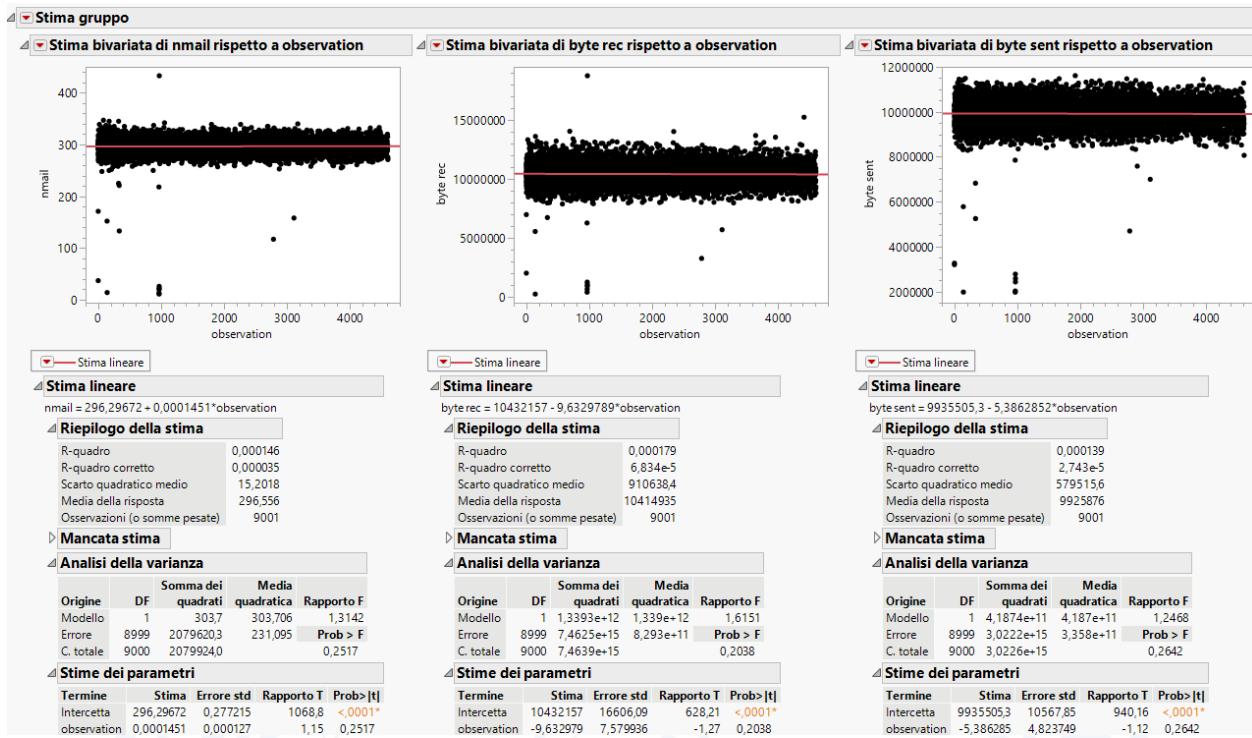


Figura 6.2: Stima lineare e valutazione R^2

6.2.2 Test di Normalità

Per decidere se utilizzare un test parametrico o non parametrico nella scelta del test statistico, è essenziale esaminare la distribuzione dei residui delle variabili coinvolte. In altre parole, la validità dell'ipotesi di normalità dei residui nelle procedure di inferenza statistica suggerisce che le discrepanze tra i valori previsti dal modello e quelli osservati nei dati seguano una distribuzione normale. Se ciò non accade, potrebbe essere opportuno considerare l'uso di un test statistico non parametrico, come ad esempio il test di Mann-Kendall. Un modo per eseguire questa valutazione è attraverso l'analisi visiva tramite dei QQplot.

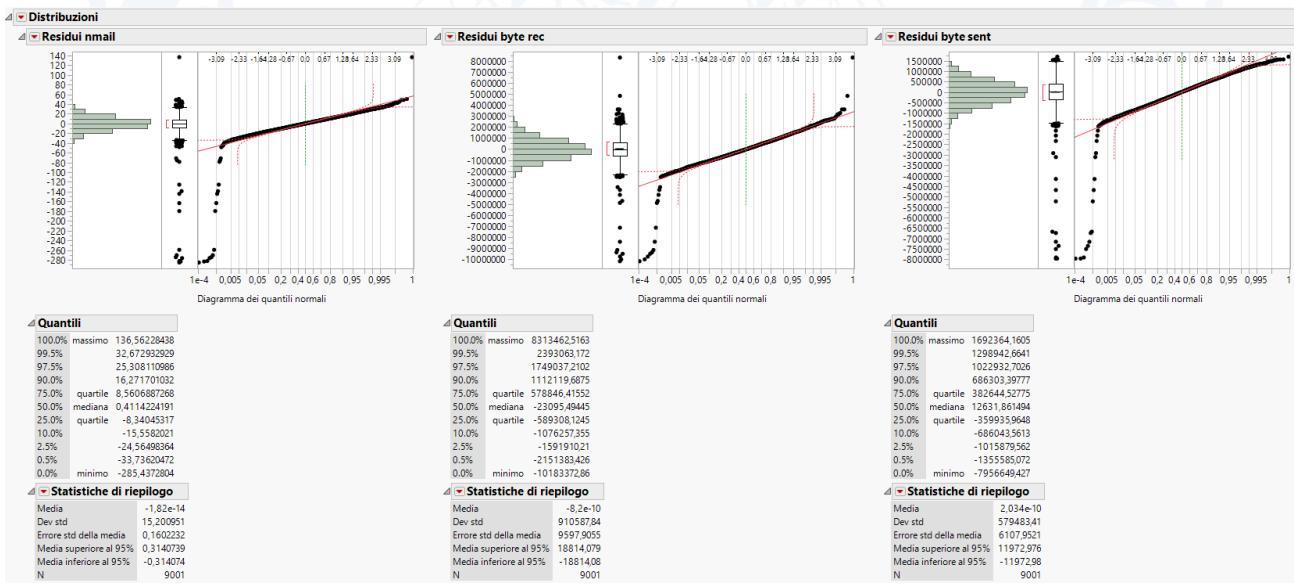


Figura 6.3: Valutazione Normalità

In tutti e 3 i casi le distribuzioni non sono normali, a questo punto dovrebbe essere verificata l'omoschedasticità, ma essendo stata rifiutata l'assunzione di normalità degli errori, è risultato superfluo effettuare tale verifica. Si valuta dunque il test di Mann-Kendall (test non parametrico).

6.2.3 Test di Mann-Kendall

Il test non parametrico di Mann-Kendall calcola il coefficiente di correlazione τ e un p-value. I valori di τ vanno da -1 a 1. Se il valore è negativo indica che le variabili sono inversamente correlate, ovvero che quando una variabile aumenta, l'altra diminuisce. I valori positivi indicano invece che quando una variabile aumenta, aumenta anche l'altra (diretta proporzionalità). **L'ipotesi nulla:**

H_0 : non è presente un trend monotono nella distribuzione.

Se il p-value è inferiore o uguale a 0,05 significa che il risultato ottenuto è statisticamente significativo per l'analisi.

Pertanto l'ipotesi è rigettata.

I risultati:

Non parametrico: τ di Kendall							
Variabile	Variabile by	τ di Kendall	Prob> τ	-,8	-,6	-,4	-,2
observation	byte rec	-0,0122	0,0834				

Figura 6.4: Kendall - Bytere

Non parametrico: τ di Kendall							
Variabile	Variabile by	τ di Kendall	Prob> τ	-,8	-,6	-,4	-,2
observation	byte sent	-0,0146	0,0383*				

Figura 6.5: Kendall - Bytesent

Non parametrico: τ di Kendall							
Variabile	Variabile by	τ di Kendall	Prob> τ	-,8	-,6	-,4	-,2
observation	nmail	-0,0004	0,9509				

Figura 6.6: Kendall - nmail

Si indicano i valori in tabella.

Test NON parametrico – Kendall – Dataset 1 – Mail Server			
	τ	p-value	H
Observation – NMail	-0.0004	0.9509	0
Observation – byte_sent	-0.0146	0.0383	1
Observation – byte_rec	-0.0122	0.0834	0

Dalla tabella l'unica variabile che presenta un trend è bytes_sent, mentre le altre due variabili non presentano trend significativi.

6.2.4 Pendenza e intercetta della retta di regressione

Si valuta, dunque, la pendenza e l'intercetta della retta di regressione utilizzando la **procedura di Theil-Sen**. Questa procedura è una **regressione lineare robusta progettata per fornire risultati più precisi e affidabili in presenza di valori di outlier nei dati**.

Per il calcolo dei valori si è utilizzato uno script Python, facendo uso della funzione theilslopes fornita dalla libreria SciPy:

Regressione Lineare Robusta – Theil e Sen

Observation (x) – byte_sent (y)

Intercetta	9954543.874080556
Pendenza	-9.667414291365773
CI inferiore	-18.762444553967473
CI superiore	-0.516734693877551

Dalla Tabella, si può notare che l'intervallo di confidenza al 95% relativo alla pendenza della retta di regressione non include lo zero, dunque, possiamo affermare che la pendenza, relativa al **modello della popolazione**, ha una probabilità del 95% che sia significativamente diversa da zero.

L'intercetta del modello della popolazione sarà valutata a valle del calcolo della pendenza. Lo stesso procedimento verrà applicato a tutte le valutazioni successive.

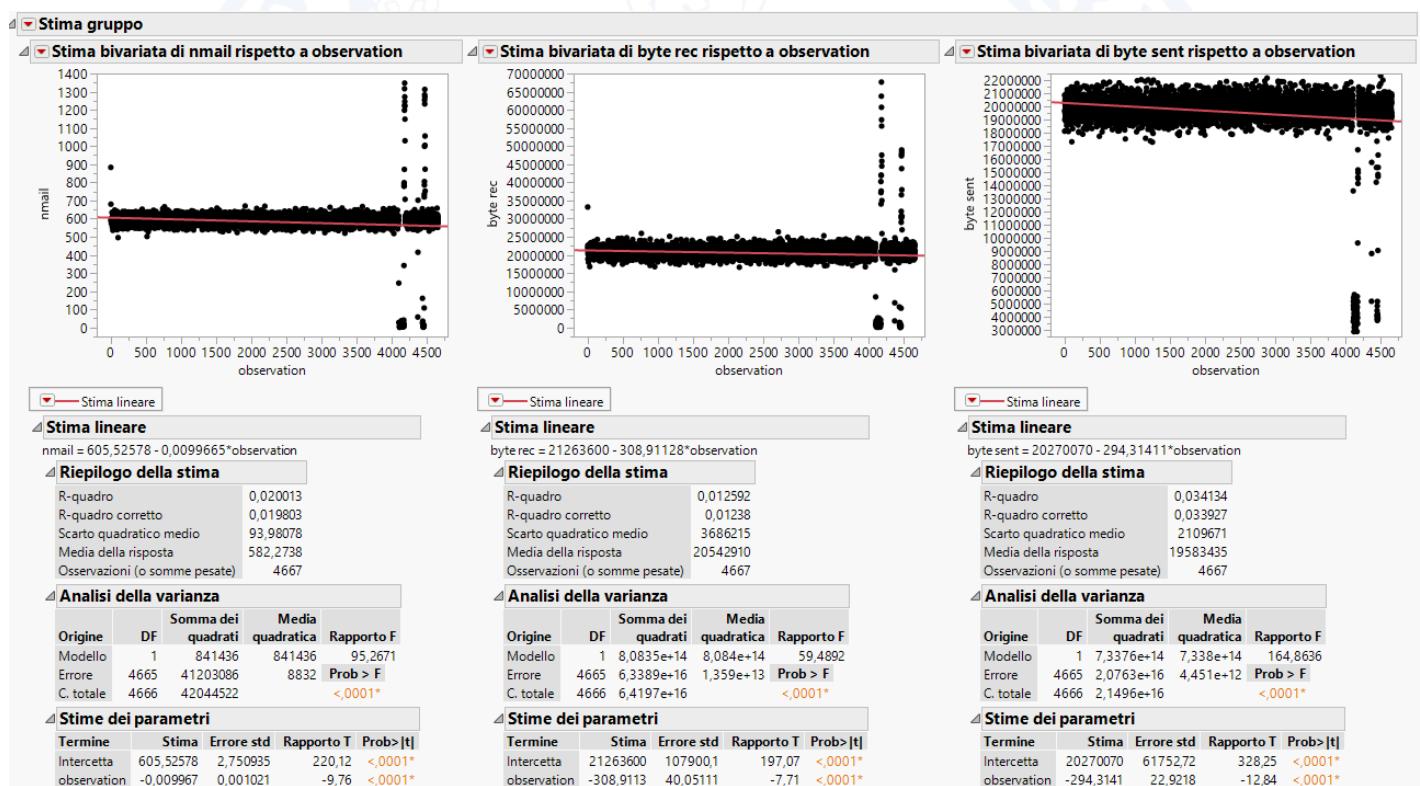
6.3 Dataset Mail Server 2

Il dataset è così formato:

- ❖ Variabili di risposta:
 - Nbyte
 - Byte_rec
 - Byte_sent
- ❖ Variabili di predizione
 - Observation

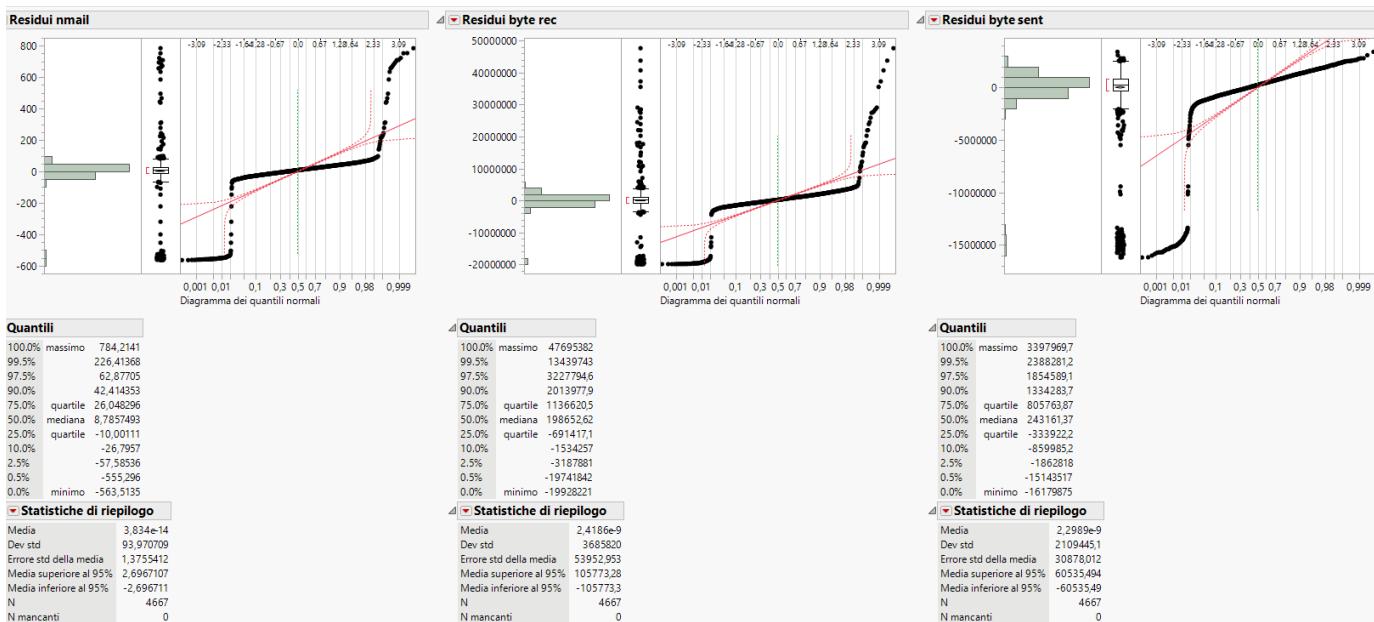
6.3.1 Valutazione delle rette di regressione e R²

Viene utilizzato un modello di regressione lineare, in cui la risposta è una funzione lineare dei predittori. Attraverso l'utilizzo del software JMP, si valuta la stima lineare delle variabili, come mostrato nella prossima Figura. Nella stima vengono forniti anche i valori di R-quadrato. Tuttavia, in questo caso, tutti e tre i valori assumono un valore molto basso, evidenziando che il modello lineare non è in grado di adattarsi correttamente ai dati. Ciò influenza negativamente sulla sua capacità predittiva.



6.3.2 Test di Normalità

Anche in questa circostanza, la selezione del tipo di test da applicare è stata basata sull'analisi della distribuzione dei residui delle rispettive variabili. In particolare, è stato condotto un test visivo utilizzando i QQplot.



In tutti e tre i casi, le distribuzioni non seguono una distribuzione normale, pertanto non viene eseguita l'analisi dell'omoschedasticità. Di conseguenza, si procede con l'applicazione del test di Mann-Kendall, che è un test non parametrico.

6.3.3 Test di Mann-Kendall

Si ripete lo stesso procedimento fatto nel primo paragrafo, ottenendo i seguenti risultati:

Non parametrico: τ di Kendall						
Variabile	Variabile by	τ di Kendall	Prob> τ	-,8	-,6	-,4
observation	byte rec	-0,0316	0,0012*			
Non parametrico: τ di Kendall						
Variabile	Variabile by	τ di Kendall	Prob> τ	-,8	-,6	-,4
byte sent	observation	-0,0392	<,0001*			
Non parametrico: τ di Kendall						
Variabile	Variabile by	τ di Kendall	Prob> τ	-,8	-,6	-,4
observation	nmail	-0,0242	0,0138*			

Si indicano i valori in tabella.

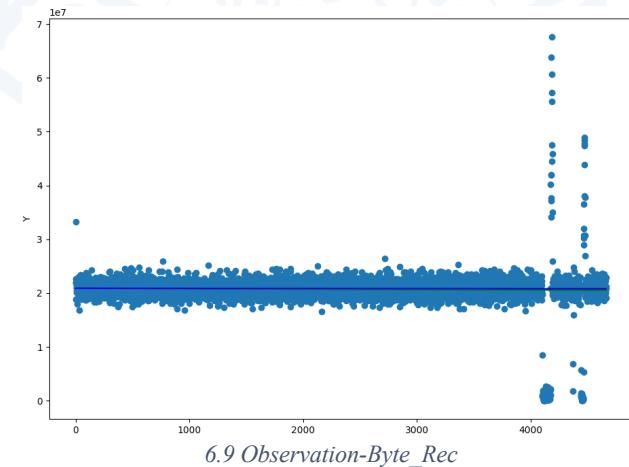
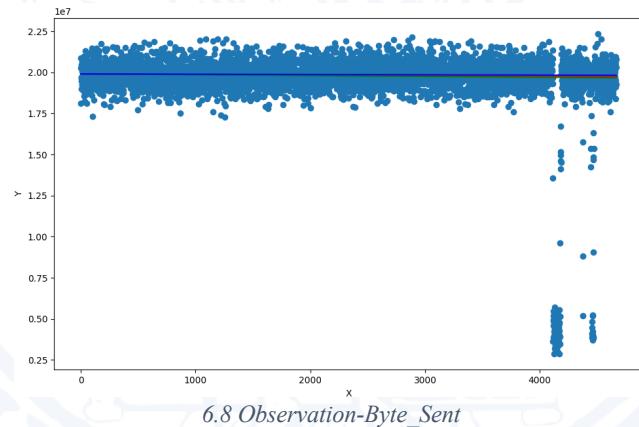
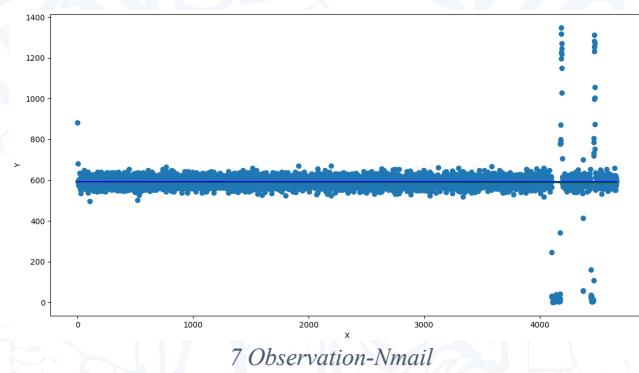
Test NON parametrico – Kendall – Dataset 2 – Mail Server			
	τ	p-value	H
Observation – NMail	-0,0242	0,0138	1
Observation – byte_sent	-0,0392	0,0001	1
Observation – byte_rec	-0,0316	0,0012	1

Dalla tabella tutti e 3 i test indicano che è presente un trend significativo nei dati.

6.3.4 Pendenza e intercetta della retta di regressione

	Regressione Lineare Robusta – Theil e Sen		
	Observation (x) – Nmail (y)	Observation (x) – byte_sent (y)	Observation (x) – byte_rec(y)
Intercetta	592.3936678614098	19903064.10051782	20890131.991364423
Pendenza	-0.0005973715651135006	-34.06305208650625	-48.29360967184802
CI inferiore	-0.0011248593925759281	-50.672672672672675	-77.54773269689737
CI superiore	0.0	-17.374087591240876	-18.993146773272414

Dalla tabella, emerge che gli intervalli di confidenza al 95% relativi alle pendenze delle rette di regressione per le variabili "byte sent" e "byte rec" non includono lo zero. Pertanto, si può affermare con una probabilità del 95% che tali pendenze, riferite al modello della popolazione, siano significativamente diverse da zero. Al contrario, per la pendenza della retta di regressione associata alla variabile "NMail", si osserva che l'intervallo di confidenza al 95% contiene lo zero. Di conseguenza, è possibile affermare con una probabilità del 95% che questa pendenza sia significativamente uguale a zero.



6.4 Confronto tra Mail Server 1 e Mail Server 2

Dall'analisi condotta sui due dataset emerge che il secondo presenta un numero maggiore di trend rispetto al primo. Ciò potrebbe indicare cambiamenti significativi nei dati del secondo dataset rispetto al primo. In generale, è importante valutare i dati e le tendenze nel contesto d'interesse, effettuando ulteriori analisi per comprendere la causa del maggiore trend e le implicazioni per l'attività in esame.

Nel nostro caso, stiamo valutando un Mail Server in relazione al trend nei dati. Per fare ciò, possiamo utilizzare diverse metriche:

- ❖ Volume delle email: un trend crescente nel numero delle email (NMail) potrebbe indicare un aumento dell'utilizzo del server, implicando la necessità di una maggiore capacità di elaborazione e archiviazione. Pertanto, effettuare valutazioni sul secondo dataset risulterebbe più significativo.
- ❖ Tasso di errore: un trend crescente nel tasso di errore (aumento della differenza tra byte-sent e byte-rec) potrebbe indicare problemi con la consegna delle email, dovuti a problemi tecnici o a indirizzi email non validi. Anche in questo caso, le valutazioni sul secondo dataset sono da preferire.
- ❖ Tasso di bounce: un trend crescente nel tasso di bounce (diminuzione dei byte-rec rispetto ai byte-sent) potrebbe indicare problemi con la consegna delle email, causati da problemi con i server di destinazione o indirizzi email non validi. Pertanto, il secondo dataset risulta più significativo.

In definitiva, grazie all'analisi del trend dei dati effettuata sui due dataset, si è in grado di determinare quale dataset risulta più significativo per le predizioni. In particolare, in relazione a alcune delle metriche considerate per valutare un Mail Server, l'importanza di ciascuna dipenderà dalle esigenze specifiche del contesto operativo.

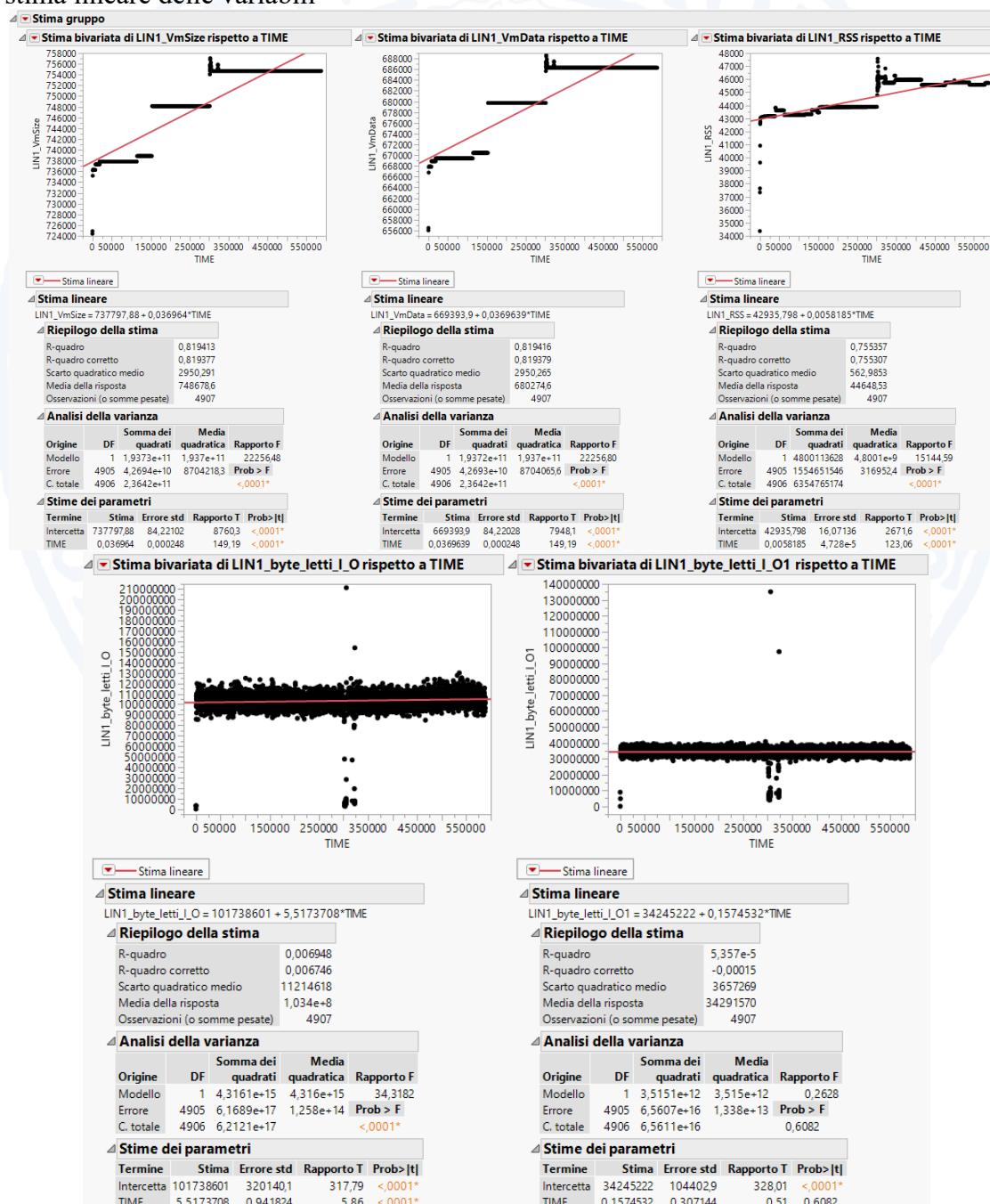
6.5 Dataset – OS1

Il dataset è così formato:

- ❖ Variabili di risposta:
 - VmSize
 - VmData
 - RSS
 - Byte_letti_IO
 - Byte_letti_IO1
- ❖ Variabili di Predizione
 - Time

6.5.1 Valutazione delle rette di regressione e R²

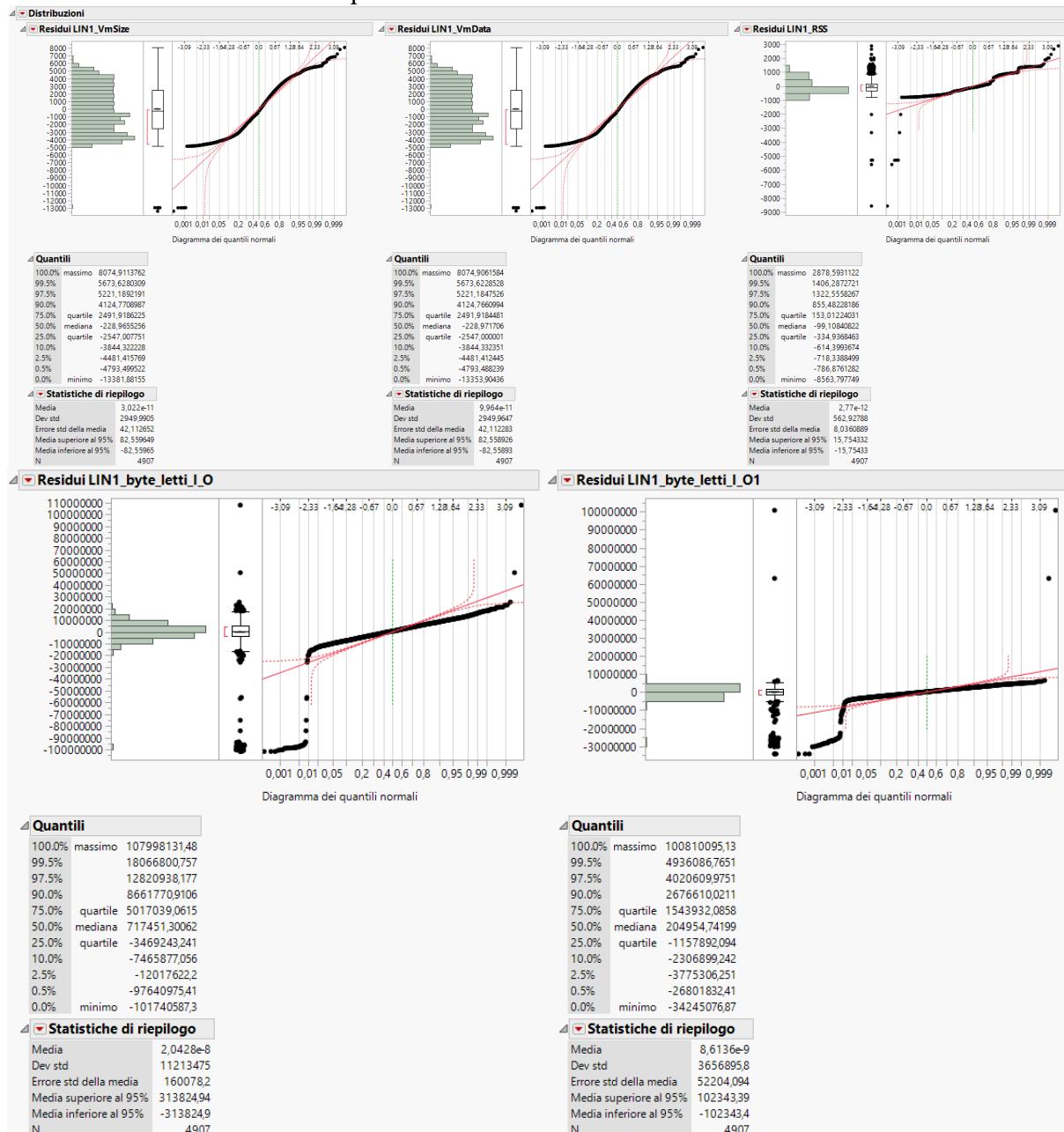
Viene applicato il modello regressivo lineare la cui risposta è una funzione lineare del predittore. Si valuta su JMP la stima lineare delle variabili



Nella stima viene indicato l'R-quadro. In questo caso R-quadro assume valore basso per le variabili ByteLetti IO e ByteLetti IO1 e alto per le altre 3 variabili.

6.5.2 Test di Normalità

Nella scelta del test d'ipotesi per valutare la presenza di trend nei dati, è fondamentale esaminare la distribuzione dei residui delle variabili coinvolte. In particolare, si può utilizzare il QQplot come test visivo per valutare l'allineamento dei dati rispetto al modello teorico.



In tutti e 5 i casi le distribuzioni non sono normali, pertanto si omette la verifica dell'omoschedasticità. Si valuta dunque il test di Mann-Kendall (test non parametrico).

6.5.3 Test di Mann-Kendall

Si ripete lo stesso procedimento fatto nel primo paragrafo ottenendo i seguenti risultati:

Non parametrico: τ di Kendall						
Variabile	Variabile by	τ di Kendall	Prob> τ	-,8	-,6	-,4
TIME	LIN1_VmData	0,8076	<,0001*			

Figura 6.10: Kendall – VmData

Non parametrico: τ di Kendall						
Variabile	Variabile by	τ di Kendall	Prob> τ	-,8	-,6	-,4
TIME	LIN1_VmSize	0,8076	<,0001*			

Figura 6.11: Kendall - VmSize

Non parametrico: τ di Kendall						
Variabile	Variabile by	τ di Kendall	Prob> τ	-,8	-,6	-,4
TIME	LIN1_RSS	0,6827	<,0001*			

Figura 6.12: Kendall – RSS

Non parametrico: τ di Kendall						
Variabile	Variabile by	τ di Kendall	Prob> τ	-,8	-,6	-,4
TIME	LIN1_byte_letti_I_O	0,0823	<,0001*			

Figura 6.13: Kendall - Byte-letti-I-O

Non parametrico: τ di Kendall						
Variabile	Variabile by	τ di Kendall	Prob> τ	-,8	-,6	-,4
TIME	LIN1_byte_letti_I_O1	-0,0017	0,8555			

Figura 6.14: Kendall - Byte-letti-I-O

Si indicano i valori in tabella e la rispettiva scelta:

Test NON parametrico – Kendall – Dataset 1 - VM			
	τ	p-value	H
Time - VmSize	0.8076	0.0001	1
Time – VmData	0.8076	0.0001	1
Time – RSS	0.6827	0.0001	1
Time – Byte_letti_I_O	0.0823	0.0001	1
Time – Byte_letti_I_O1	-0.0017	0.8555	0

Dalla tabella si evince che 4 test su 5 indicano che hanno un trend significativo nei dati.

6.5.4 Pendenza e intercetta della retta di regressione

Viene ripetuto lo stesso procedimento del paragrafo 1:

Regressione Lineare Robusta – Theil e Sen				
	Time(x) – VmSize(y)	Time(x) – VmData(y)	Time(x) – RSS(y)	Time(x) – Byte_letti_I_O(y)
Intercetta	738854.3947841726	670450.3947841726	42428.80701754386	102763229.503125
Pendenza	0.03157224220623501	0.03157224220623501	0.004970760233918129	4.737578125
CI Inferiore	0.030715952172645087	0.030715952172645087	0.004876649454962708	3.6739155483258776
CI superiore	0.03232097662647015	0.03232097662647015	0.005055788005578801	5.80459886547812

Dalla tabella, emerge che gli intervalli di confidenza al 95% associati alle pendenze delle rette di regressione non contengono lo zero per nessuna delle variabili considerate. Pertanto, è possibile affermare con un livello di confidenza del 95% che le pendenze, riferite al modello della popolazione, sono statisticamente diverse da zero.

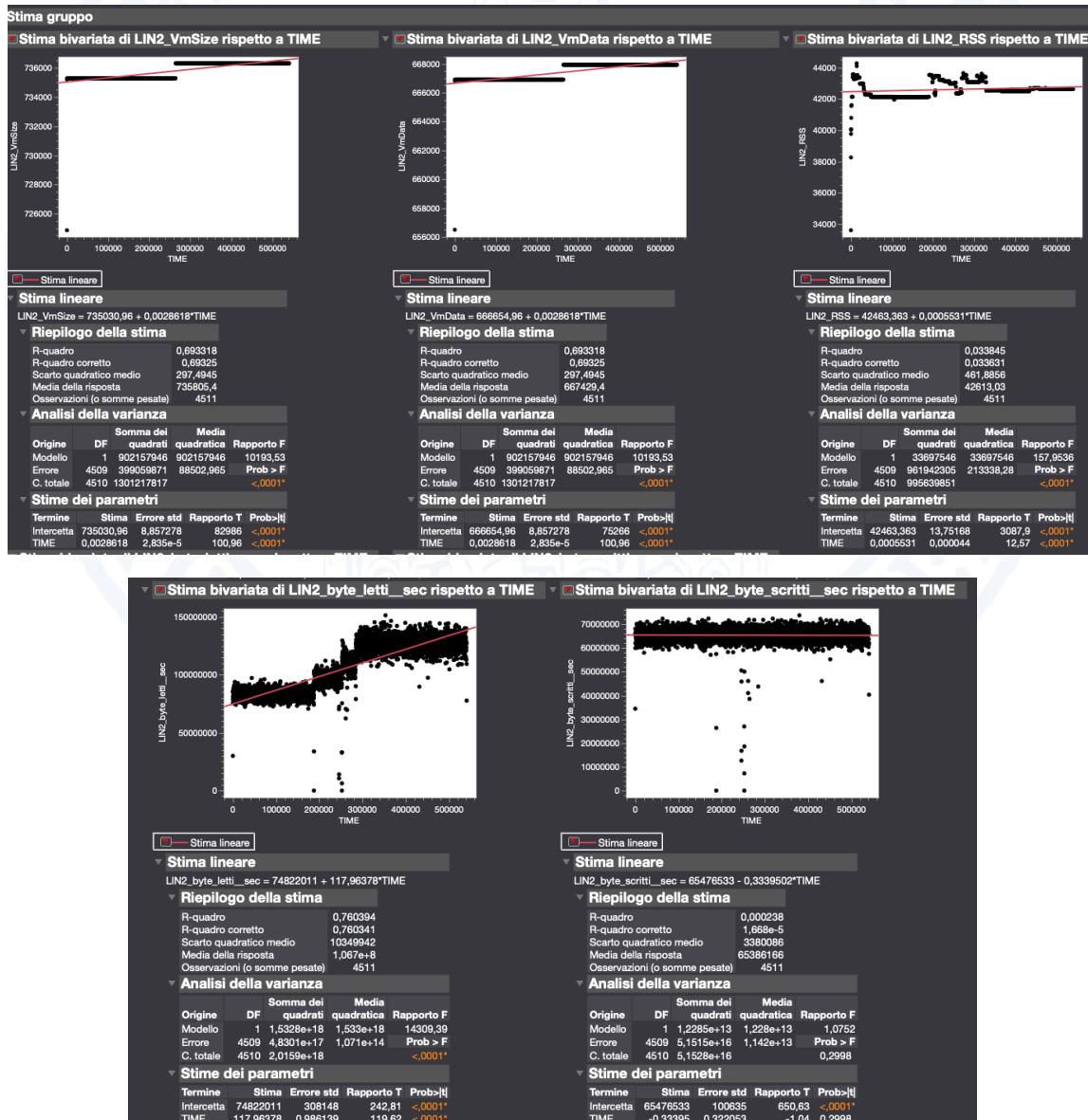
6.6 Dataset – OS2

Il dataset è così composto:

- ❖ Variabili di risposta:
 - VmSize
 - VmData
 - RSS
 - Byte letti al secondo
 - Byte scritti al secondo
- ❖ Variabili di predizione:
 - Time

6.6.1 Valutazione delle rette di regressione e R²

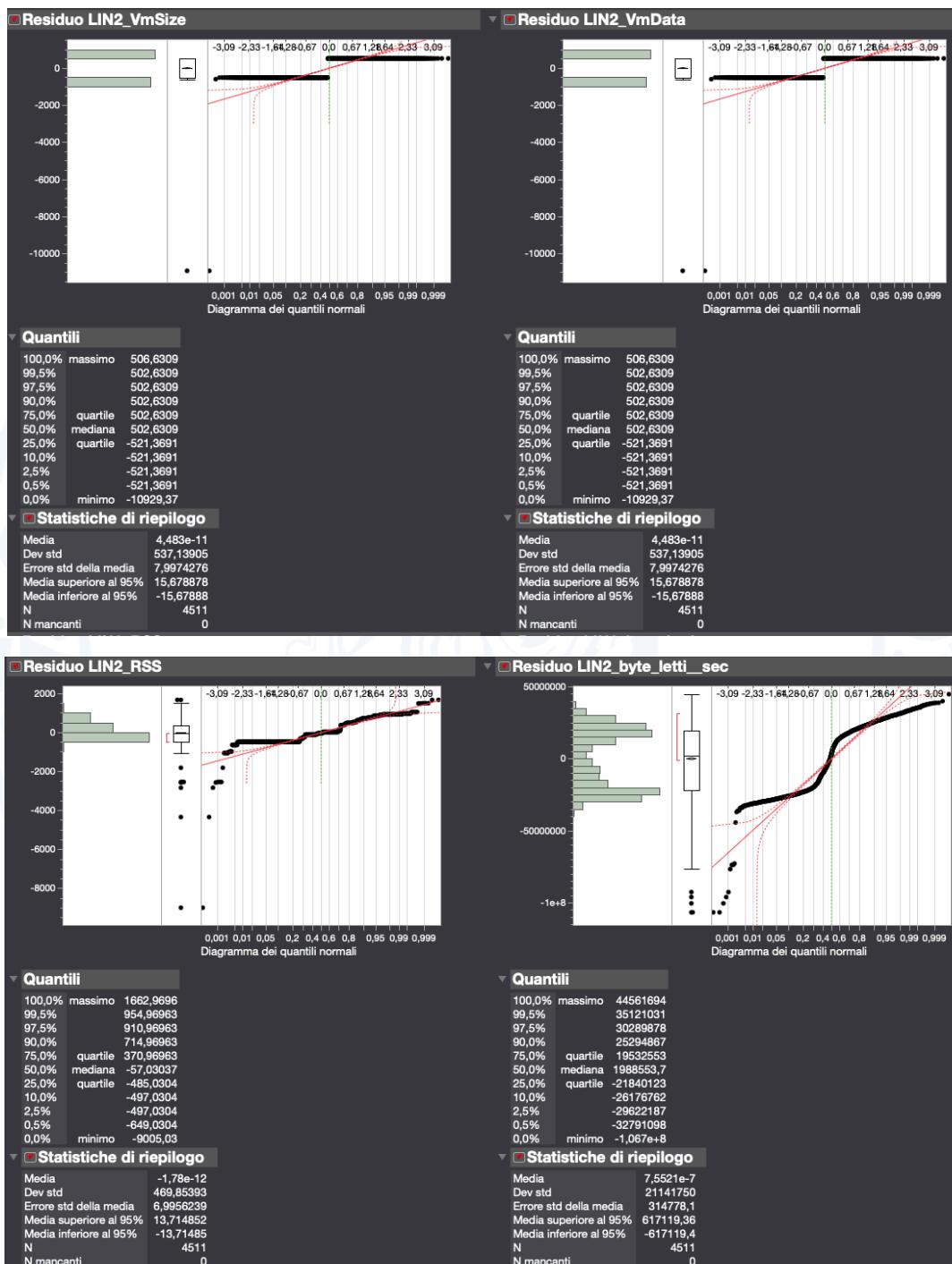
Viene utilizzato un modello di regressione lineare in cui la variabile dipendente è una funzione lineare delle variabili predittive. La stima lineare delle variabili è valutata tramite il software JMP, come indicato nella figura.

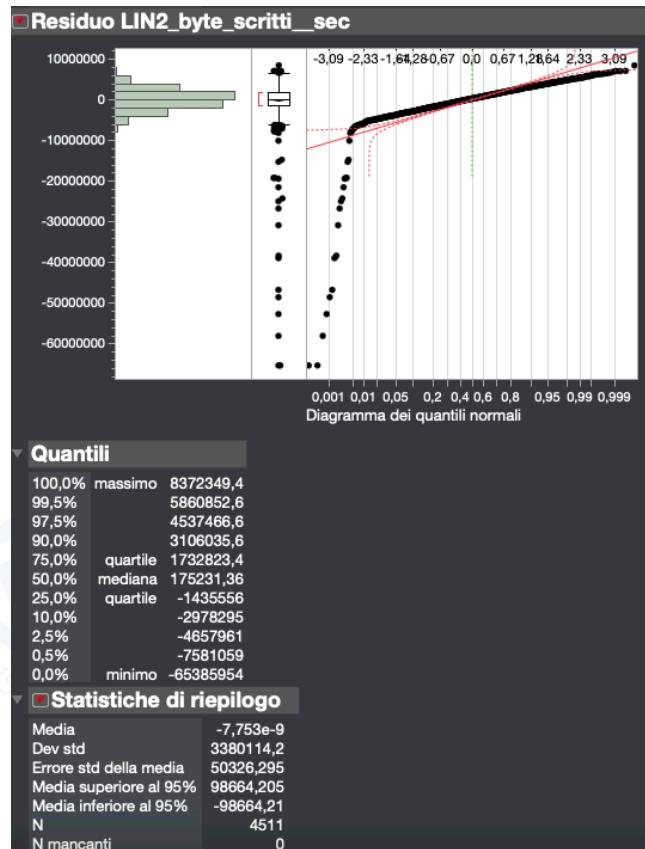


Nella stima, l'indicatore di bontà di adattamento R-quadrato è mostrato in figura. In questo caso, il valore di R-quadrato è basso per le variabili "Byte scritti al secondo" e "RSS", mentre è elevato per le altre tre variabili.

6.6.2 Test di Normalità

Per determinare il test appropriato per verificare la presenza di trend nei dati è stata valutata la distribuzione dei residui delle rispettive variabili. In particolare, è stato eseguito un test visivo utilizzando i QQplot.





In tutti e cinque i casi, le distribuzioni non seguono una distribuzione normale, pertanto si omette l'analisi dell'omoschedasticità. Si procede quindi con l'applicazione del test di Mann-Kendall, che è un test non parametrico.

6.6.3 Test di Mann-Kendall

Si ripete lo stesso procedimento fatto nel primo paragrafo ottenendo i seguenti risultati:



Figura 15 Kendall - VmData



Figura 16 Kendall - VmSize



Figura 17 Kendall - RSS



Figura 18 Kendall - byte letti sec



Figura 19 Kendall - byte scritti sec

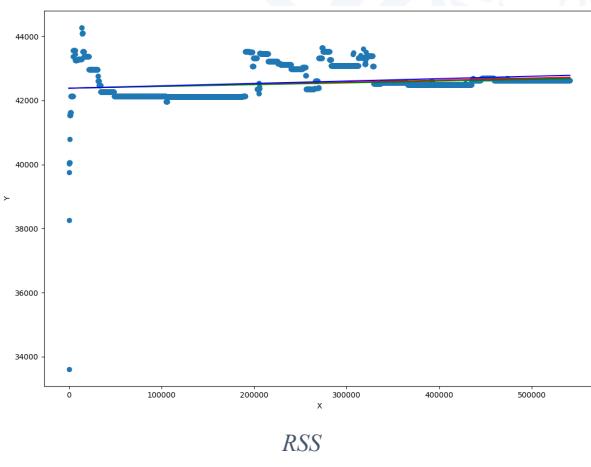
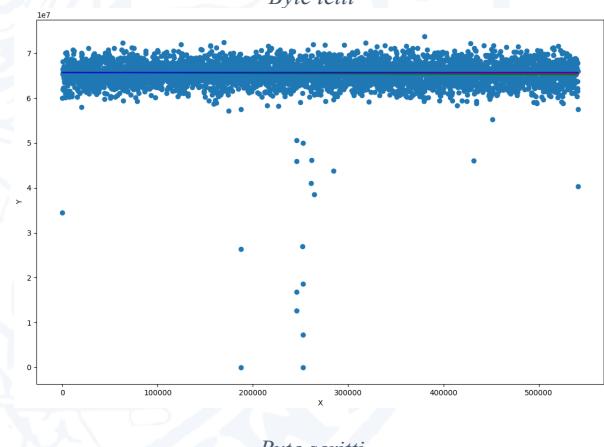
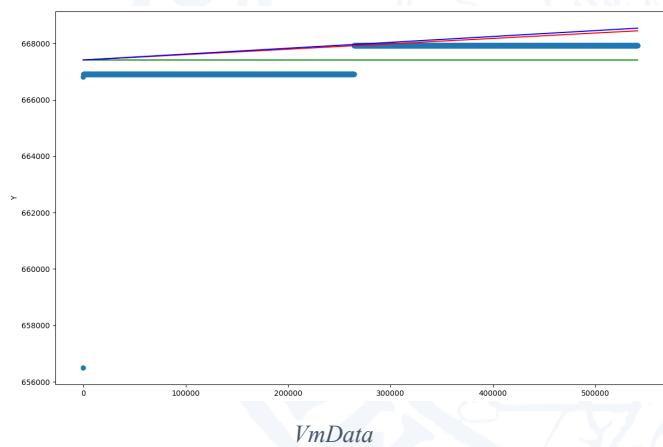
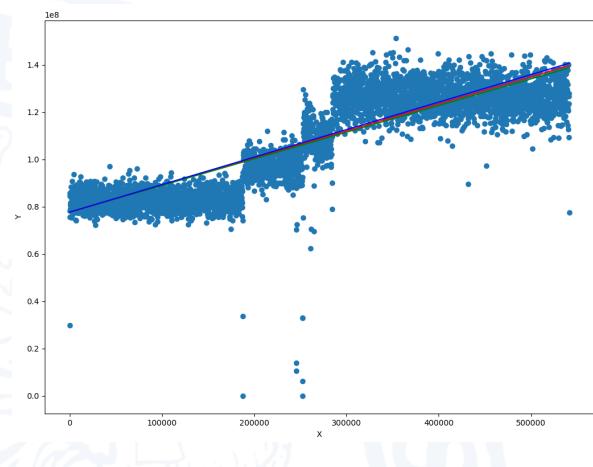
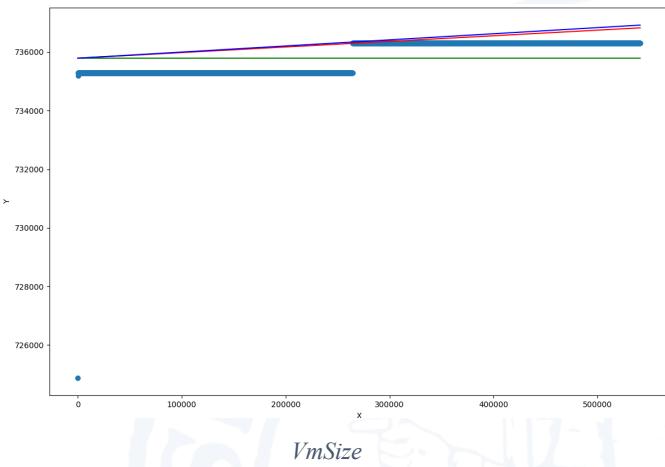
Vengono riportati in forma tabulare i valori:

Test NON parametrico – Kendall – Dataset 1 - VM			
	τ	p-value	H
Time - VmSize	0.7065	0.0001	1
Time – VmData	0.7065	0.0001	1
Time – RSS	0.1788	0.0001	1
Time – Byte_letti_sec	0.6259	0.0001	1
Time – Byte_scritti_sec	-0.0210	0.0342	1

6.6.4 Pendenza e intercetta della retta di regressione

Si ripete lo stesso procedimento del paragrafo precedente:

Regressione Lineare Robusta – Theil e Sen					
	VmSize(y)	VmData(y)	RSS(y)	Byte_letti_sec(y)	Byte_scritti_sec(y)
Intercetta	735790.3761488	667414.3761488455	42384.85453267	102763229.5031	77733384.23784
Pendenza	0.0019128745423	0.00191287454232	0.00063246661	4.737578125	114.5535098
CI Inferiore	0.0	0.0	0.00055555555	3.6739155483258	112.73359327217125
CI superiore	0.002085369827	0.002085369827	0.000744544287	5.80459886547	116.37659764826176



Dalla tabella, è evidente che gli intervalli di confidenza al 95% relativi alle pendenze delle rette di regressione per le variabili "byte letti al secondo", "byte scritti al secondo" e "RSS" non includono lo zero. Di conseguenza, si può affermare con un livello di confidenza del 95% che tali pendenze, riferite al modello della popolazione, siano significativamente diverse da zero. D'altra parte, per le pendenze delle rette di regressione relative alle variabili "VmSize" e "VmData", si osserva che gli intervalli di confidenza al 95% contengono lo zero. Pertanto, si può affermare con un livello di confidenza del 95% che tali pendenze siano significativamente uguali a zero.

6.7 Dataset – OS3

Il dataset è così formato:

- ❖ Variabili di risposta:

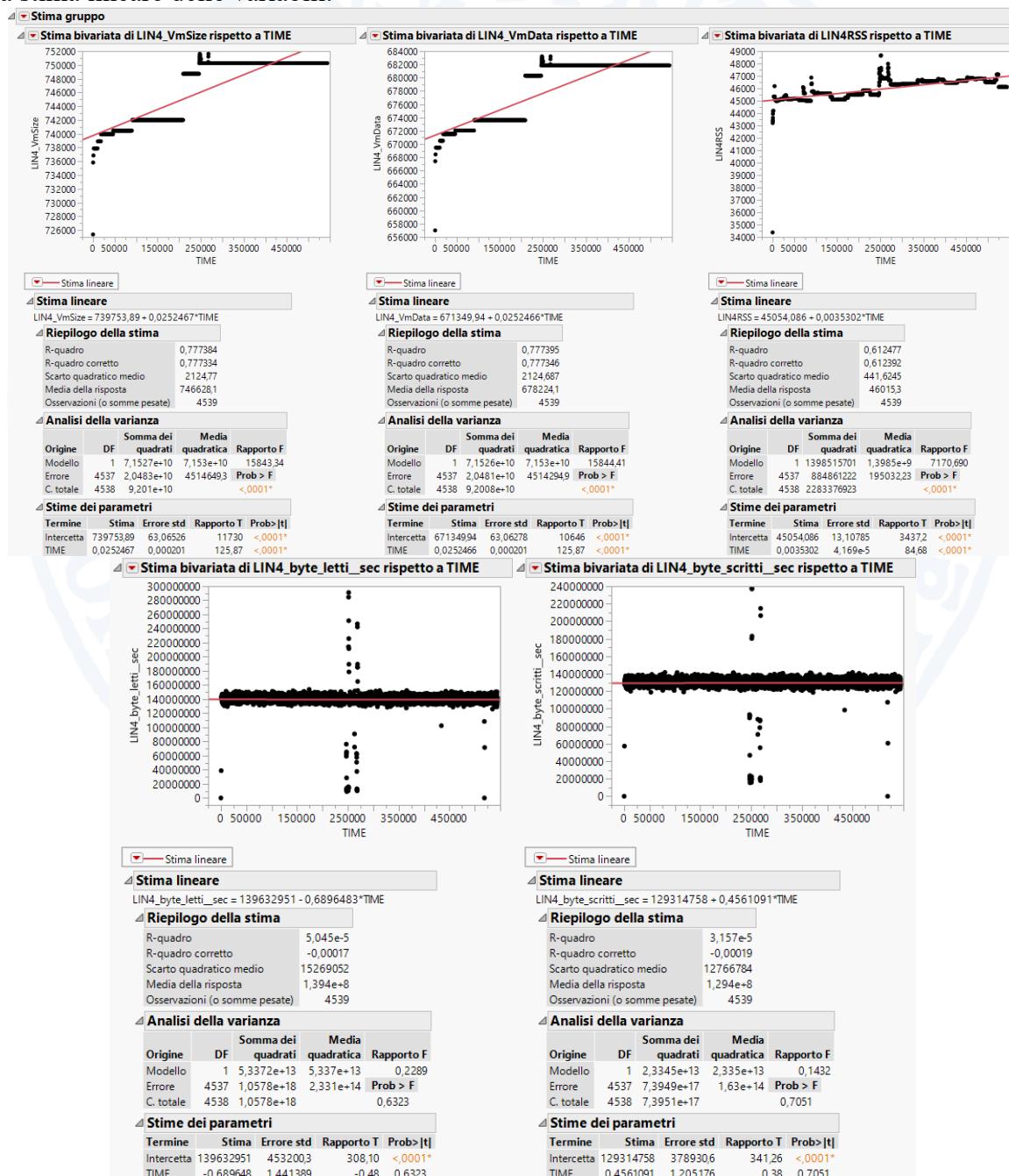
- VmSize
- VmData
- RSS
- Byte_letti_sec
- Byte_scritti_sec

- ❖ Variabili di predizione

- Time

6.7.1 Valutazione delle rette di regressione e R²

Viene applicato il modello regressivo lineare la cui risposta è una funzione lineare del predittore. Si valuta su JMP la stima lineare delle variabili.

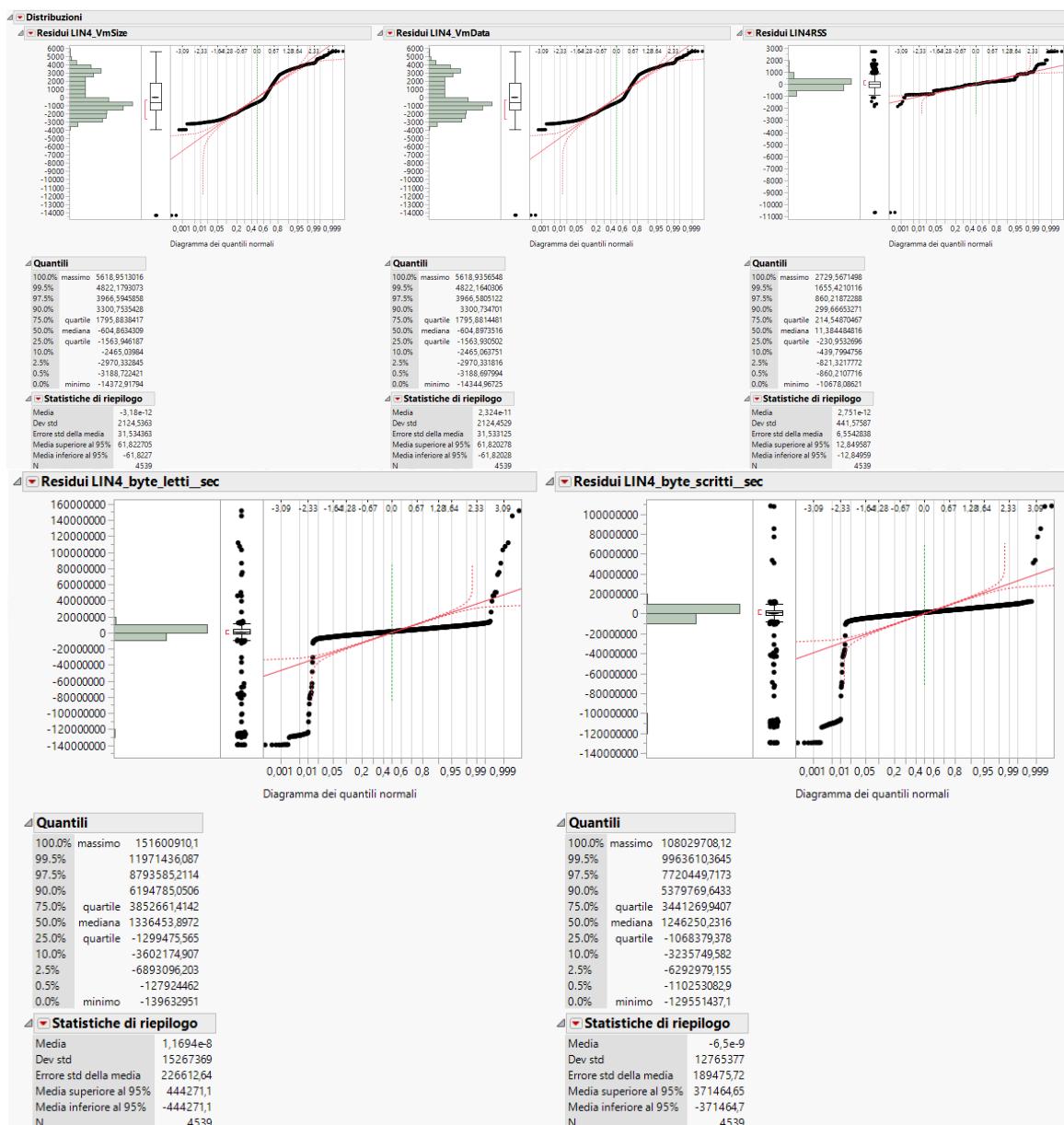


Nella stima viene indicato l'R-quadro. In questo caso R-quadro assume valore basso per le variabili Byte letti sec e Byte scritti sec e alto per le altre 3 variabili.

6.7.2 Test di Normalità

Per la scelta del tipo di test d'ipotesi da eseguire è stata valutata la distribuzione dei residui delle rispettive variabili.

In particolare, si applica il test visivo tramite dei QQplot.



In tutti e 5 i casi le distribuzioni non sono normali, pertanto si omette di verificare l'omoschedasticità dei residui. Si valuta dunque il test di Mann-Kendall (test non parametrico).

6.7.3 Test di Mann-Kendall

Si ripete lo stesso procedimento visto nel primo paragrafo ottenendo i seguenti risultati:

▼ Non parametrico: τ di Kendall

Variabile	Variabile by	τ di Kendall	Prob> τ
LIN4_VmData	TIME	0,7832	<,0001*

▼ Non parametrico: τ di Kendall

Variabile	Variabile by	τ di Kendall	Prob> τ
LIN4_VmSize	TIME	0,7832	<,0001*

▼ Non parametrico: τ di Kendall

Variabile	Variabile by	τ di Kendall	Prob> τ
LIN4RSS	TIME	0,5938	<,0001*

▼ Non parametrico: τ di Kendall

Variabile	Variabile by	τ di Kendall	Prob> τ
LIN4_byte_letti_sec	TIME	-0,0506	<,0001*

▼ Non parametrico: τ di Kendall

Variabile	Variabile by	τ di Kendall	Prob> τ
LIN4_byte_scritti_sec	TIME	-0,0171	0,0841

Vengono riportati i valori in tabella con la rispettiva scelta:

Test NON parametrico – Kendall – Dataset 3 – VM4			
	τ	p-value	H
Time - VmSize	0.7832	0.0001	1
Time – VmData	0.7832	0.0001	1
Time – RSS	0.5938	0.0001	1
Time – Byte_letti_sec	-0.0506	0.0001	1
Time – Byte_letti_I_O1_sec	-0.0171	0.0841	0

Dalla tabella, si nota che 4 test su 5 affermano la presenza di trend significativi nei dati.

6.7.4 Pendenza e intercetta della retta di regressione

Si ripete lo stesso procedimento del paragrafo 1:

Regressione Lineare Robusta – Theil e Sen				
	Time(x) – VmSize(y)	Time(x) – VmData(y)	Time(x) – RSS(y)	Time(x) – Byte_letti_sec
Intercetta	744374.5991189427	675970.5991189427	45386.471028037384	141291080.77144608
Pendenza	0.021659324522760644	0.021659324522760644	0.0034579439252336447	-1.891048815359477
CI Inferiore	0.02127364220024293	0.02127364220024293	0.0033884297520661156	-2.617641979071421
CI superiore	0.02202594299368493	0.02202594299368493	0.0035222894881673087	-1.16775956284153

Dalla tabella si evince che gli intervalli di confidenza al 95%, associati alle pendenze delle rette di regressione, escludono lo zero per ciascuna variabile. Di conseguenza, è plausibile affermare con un livello di confidenza del 95% che le pendenze, riferite al modello della popolazione, risultino significativamente diverse da zero.

6.7.5 Confronto tra OS1-OS2-OS3

Dall'analisi del trend dei dati raccolti nei tre dataset relativi all'occupazione di memoria di una macchina virtuale (VM), è emerso che il secondo dataset presenta un numero maggiore di trend, poiché tutte le variabili di risposta mostrano una significativa tendenza. La presenza di un trend più marcato nei dati può risultare vantaggiosa in quanto indica una direzione o una tendenza nei movimenti dei dati. Tale informazione può rivelarsi preziosa per prevedere futuri cambiamenti e per prendere decisioni informate. Nel contesto di una VM, la presenza di un trend più pronunciato può essere utile per determinare se allocare o meno una maggiore quantità di memoria alla macchina virtuale.

Nel dettaglio, l'individuazione di dati con una tendenza significativa per una VM apre la possibilità di considerare diverse metriche, consentendo di adottare approcci più formali e analitici nella gestione dell'allocazione di risorse.

L'analisi dell'utilizzo della memoria suggerisce che un aumento sia nella dimensione della VM che nei dati della VM potrebbe indicare un'attività più intensa della VM, indicando la necessità di una maggiore quantità di memoria. Pertanto, le valutazioni possono essere effettuate indifferentemente su uno qualsiasi dei tre dataset, poiché tutte le variabili mostrano un trend significativo.

Per quanto riguarda lo spazio di archiviazione, un aumento nell'utilizzo dello spazio (rappresentato dall'aumento di VM RSS) potrebbe indicare un'attività più intensa della VM, suggerendo la necessità di più spazio di archiviazione. Anche in questo caso, le decisioni possono essere prese su uno qualsiasi dei tre dataset, poiché tutti sono statisticamente significativi.

Per quanto riguarda il tasso di errore, un aumento nel tasso di errore (riduzione di Byte letti al secondo rispetto a Byte scritti al secondo) potrebbe indicare problemi con la VM, come incompatibilità con il sistema operativo o problemi di configurazione. In questo caso, per prevedere andamenti futuri, è necessario considerare il secondo dataset, poiché le variabili di risposta interessate a questa metrica mostrano un trend significativo solo in quel set di dati.

6.8 VmRes1 – Failure Prediction

Il dataset è formato come segue:

Variabili di risposta:

- ❖ allocatedheap

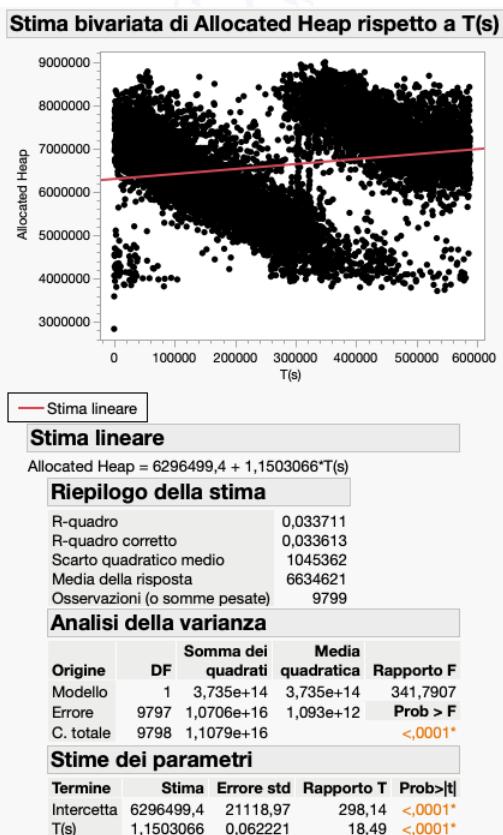
Variabili di predizione:

- ❖ Time

6.8.1 Valutazione della retta di regressione e R^2

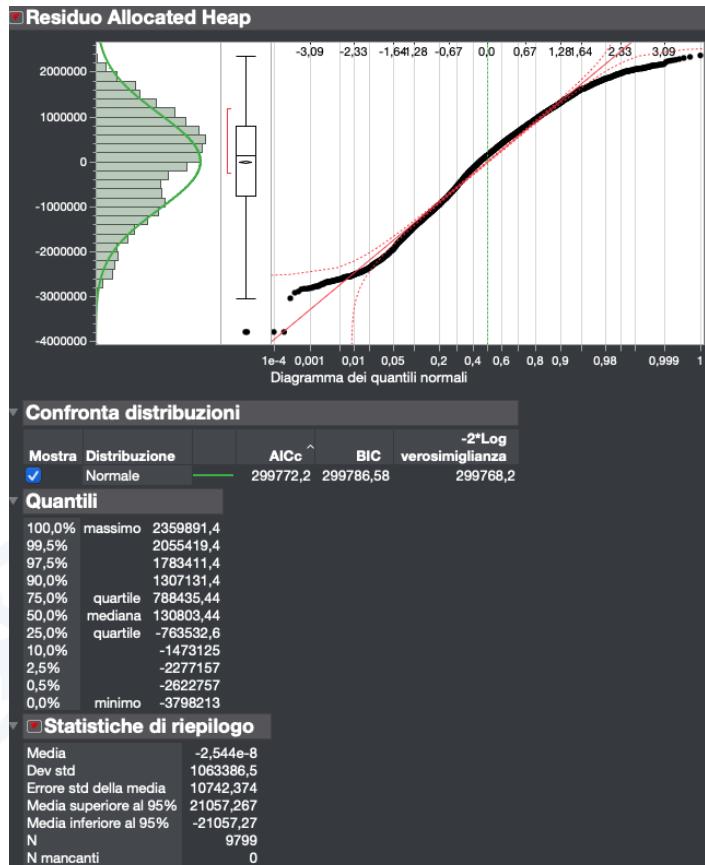
Viene adottato un modello di regressione lineare in cui la risposta è una funzione lineare del predittore. La stima lineare delle variabili è valutata tramite JMP, come evidenziato nella Figura 5.33.

Nella stima, è fornito l'indicatore di bontà di adattamento R-quadrato. In questo caso, l'R-quadrato assume un valore basso per la variabile AllocatedHeap.



6.8.2 Test di Normalità

Per la selezione del test da eseguire al fine di individuare tendenze significative nei dati, è stata valutata la distribuzione dei residui delle rispettive variabili. In particolare, è stato utilizzato un approccio visivo mediante l'analisi dei QQplot.



In questo caso, poiché la distribuzione non è normale, non viene effettuata l'analisi dell'omoschedasticità. Si procede, quindi, con l'analisi mediante il test di Mann-Kendall, che è un test non parametrico.

6.8.3 Test di Mann-Kendall

Si ripete lo stesso procedimento fatto nel primo paragrafo ottenendo il seguente risultato:

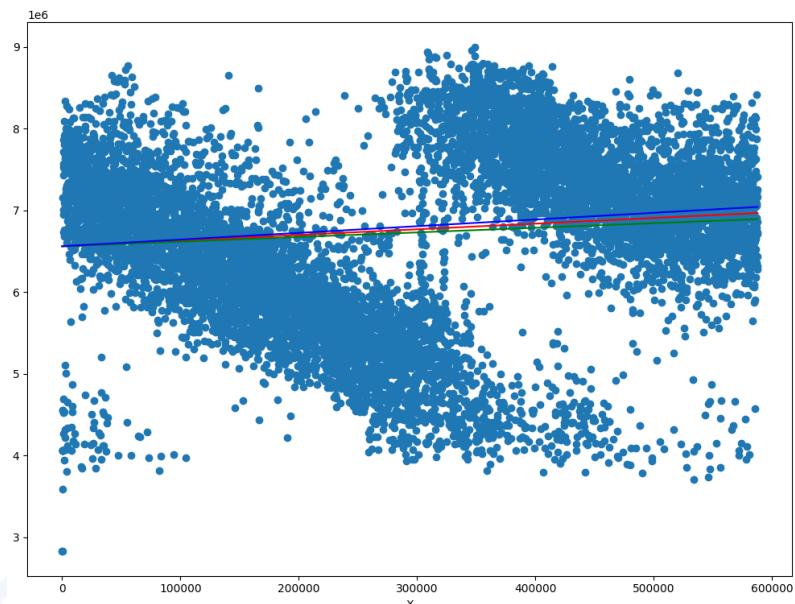
Non parametrico: τ di Kendall						
Variabile	Variabile by	τ di Kendall	Prob> τ	<0,0001*		
Allocated Heap	T(s)	0,0700				

Dalla Figura, è evidente che il valore di p-value è molto basso, il che indica che l'ipotesi H_0 è pari a 1. Ciò denota la presenza di un trend significativo nella rispettiva variabile.

6.8.4 Pendenza e intercetta della retta di regressione

Si ripete lo stesso procedimento del primo paragrafo:

Regressione Lineare Robusta – Theil e Sen – Predizione 1	
Intercetta	Ts(x) – Heap (y)
Pendenza	6562275,461669506
CI inferiore	0,6911224682945296
CI superiore	0,563777264562484
	0,8178240089963452



Dalla tabella, è evidente che l'intervallo di confidenza al 95% relativo alla pendenza della retta di regressione non contiene lo zero. Pertanto, si può affermare con un livello di confidenza del 95% che la pendenza, riferita al modello della popolazione, sia significativamente diversa da zero.

6.8.5 Valutazione Failure Prediction

La prediction della failure si calcola:

$$allocatedHeap = m + bT(s)$$

Pertanto:

$$T(s) = \frac{(1\text{ GB} - m)}{b} = \frac{(1\text{ GB} - 6562275.46)}{(0.69 \pm 0.12)} = 49 \pm 8 \text{ anni}$$

In cui l'intervallo di confidenza al 95% è stato calcolato come segue:

$$CI = \frac{CI_{superiore} - CI_{inferiore}}{2} \cong 0.12$$

6.9 VmRes2 – Failure Prediction

Il dataset è formato come segue:

Variabili di risposta:

- ❖ allocatedheap

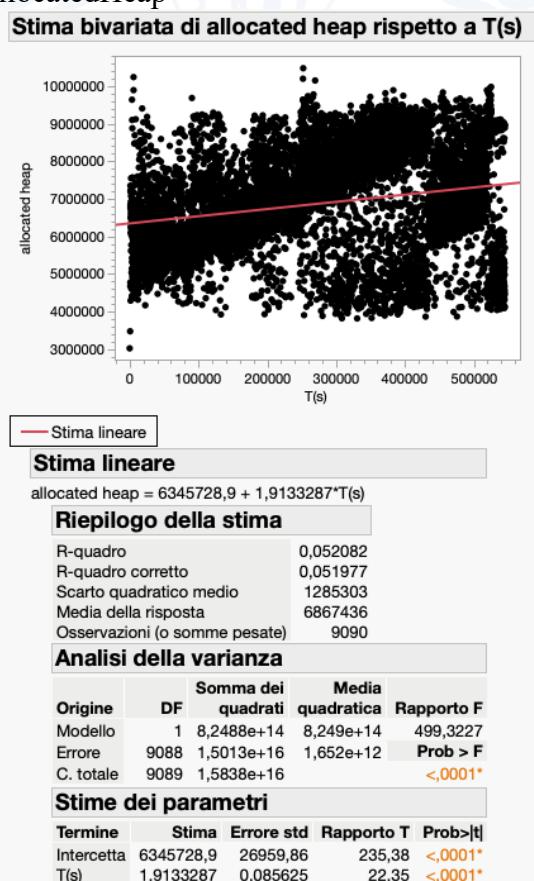
Variabili di predizione:

- ❖ Time

6.9.1 Valutazione della retta di regressione e R^2

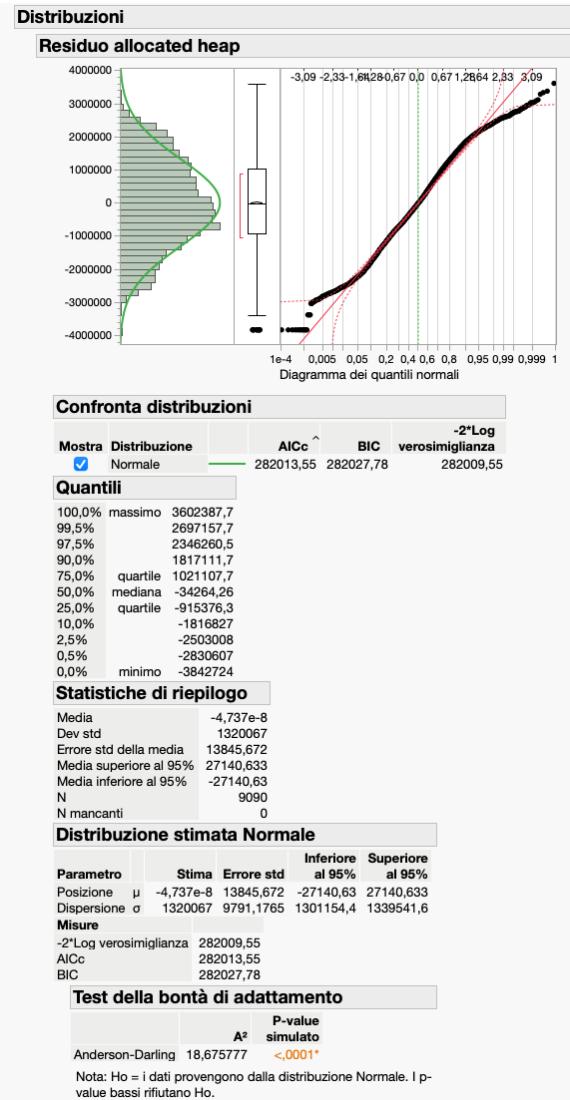
Viene adottato un modello di regressione lineare in cui la risposta è una funzione lineare del predittore. La stima lineare delle variabili è valutata tramite JMP, come evidenziato nella Figura 5.33.

Nella stima, è fornito l'indicatore di bontà di adattamento R-quadrato. In questo caso, l'R-quadrato assume un valore basso per la variabile AllocatedHeap



6.9.2 Test di Normalità

Per la selezione del test da eseguire al fine di individuare tendenze significative nei dati, è stata valutata la distribuzione dei residui delle rispettive variabili. In particolare, è stato utilizzato un approccio visivo mediante l'analisi dei QQplot.



In questo caso, poiché la distribuzione non è normale, non viene effettuata l'analisi dell'omoschedasticità. Si procede, quindi, con l'analisi mediante il test di Mann-Kendall, che è un test non parametrico.

6.9.3 Test di Mann-Kendall

Si ripete lo stesso procedimento fatto nel primo paragrafo ottenendo il seguente risultato:

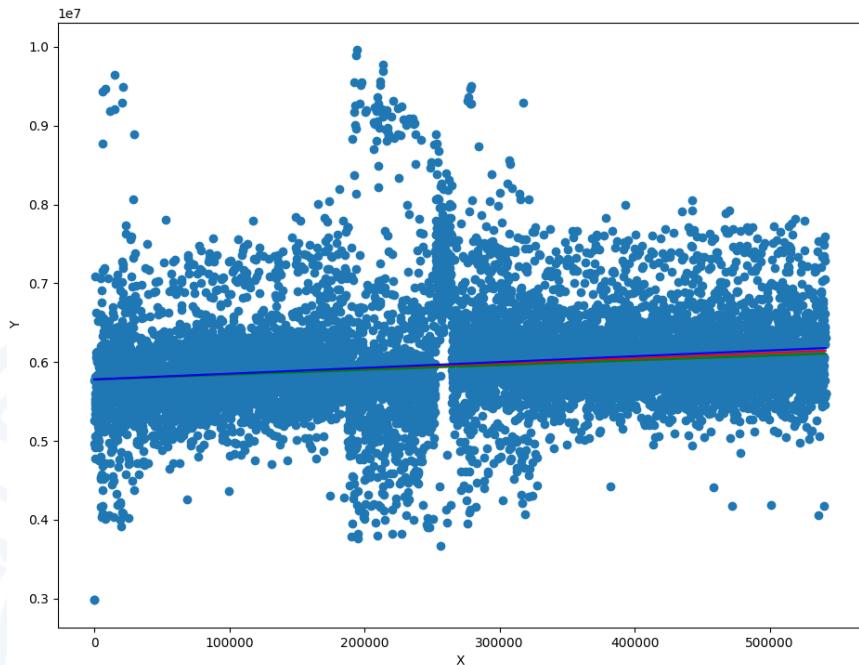
Non parametrico: τ di Kendall		
Variabile	Variabile by	τ di Kendall
allocated heap T(s)	T(s)	-0,2031 <.0001*

Dalla Figura, è evidente che il valore di p-value è molto basso, il che indica che l'ipotesi H_0 è pari a 1. Ciò denota la presenza di un trend significativo nella rispettiva variabile.

6.9.4 Pendenza e intercetta della retta di regressione

Si ripete lo stesso procedimento del paragrafo 1:

Regressione Lineare Robusta – Theil e Sen – Predizione 2	
	Ts(x) – Heap (y)
Intercetta	5780180.563324536
Pendenza	0.6709499304572171
CI inferiore	0.6048638415610893
CI superiore	0.7366538131962297



Dalla tabella, è evidente che l'intervallo di confidenza al 95% relativo alla pendenza della retta di regressione non contiene lo zero. Pertanto, si può affermare con un livello di confidenza del 95% che la pendenza, riferita al modello della popolazione, sia significativamente diversa da zero.

6.9.5 Valutazione Failure Prediction

La prediction della failure si calcola:

$$\text{allocatedHeap} = m + bT(s)$$

Pertanto:

$$T(s) = \frac{(1 \text{ GB} - m)}{b} = \frac{(1 \text{ GB} - 5780180.56)}{(0.67 \pm 0.06)} = 50 \pm 5 \text{ anni}$$

In cui l'intervallo di confidenza al 95% è stato calcolato come segue:

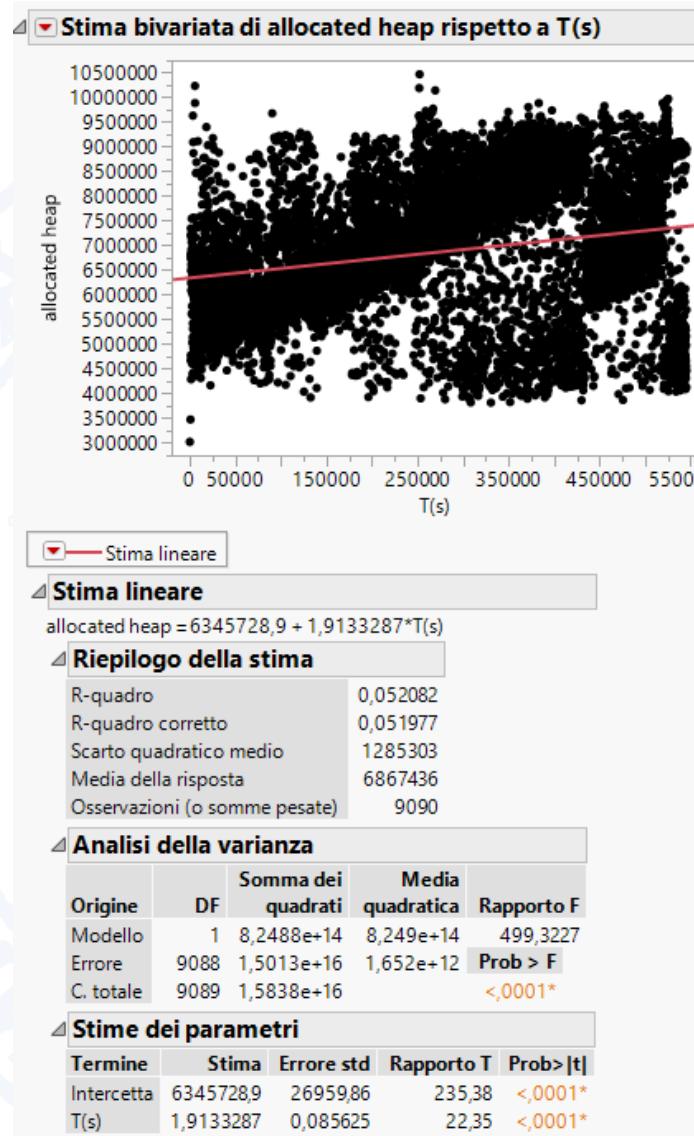
$$CI = \frac{CI_{superiore} - CI_{inferiore}}{2} \cong 0.06$$

6.10 VmRes3 – Failure Prediction

Il dataset è così formato:

- ❖ Variabili di risposta:
 - Allocatedheap
- ❖ Variabili di predizione
 - Time

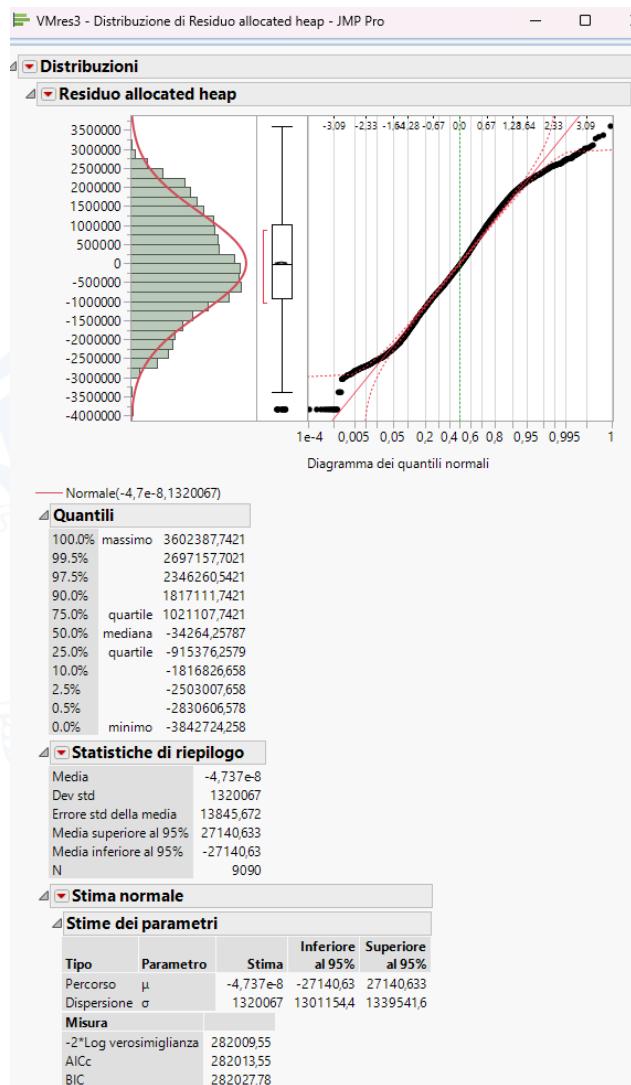
6.10.1 Valutazione delle rette di regressione e R²



Nella stima viene indicato l'R-quadro. In questo caso R-quadro assume valore basso per la variabile AllocatedHeap.

6.10.2 Test di Normalità

Per la scelta del test da eseguire è stata valutata la distribuzione dei residui delle rispettive variabili. In particolare, si applica il test visivo tramite dei QQplot.



In questo caso la distribuzione non è normale, pertanto si omette la verifica dell' omoschedasticità. Si valuta, dunque, il test di Mann-Kendall (test non parametrico).

6.10.3 Test di Mann-Kendall

Si ripete lo stesso procedimento fatto nel primo paragrafo ottenendo il seguente risultato:

Non parametrico: τ di Kendall						
Variabile	Variabile by	τ di Kendall	Prob> τ	-,8	-,6	-,4
allocated heap	T(s)	0,2031	<,0001*			

Dalla tabella si può notare che il valore di p-value è molto basso e dunque, l'ipotesi H_0 è pari a 1; ciò denota la presenza di un trend significativo nella rispettiva variabile.

6.10.4 Pendenza e intercetta della retta di regressione

Si ripete lo stesso procedimento del paragrafo 1:

Regressione Lineare Robusta – Theil e Sen – Predizione 3	
	Ts(x) – Heap (y)
Intercetta	6041585.556431387
Pendenza	2.9030932760062105
CI inferiore	2.714759535655058
CI superiore	3.0923529411764705

Dalla tabella, si osserva che l'intervallo di confidenza al 95% per la pendenza della retta di regressione non contiene lo zero. Pertanto, è possibile affermare con un livello di confidenza del 95% che la pendenza associata al modello della popolazione è statisticamente significativamente diversa da zero.

6.10.5 Valutazione Failure Prediction

La prediction della Failure è:

$$\text{allocatedheap} = m + bT(s)$$

Pertanto, dati i valori in tabella, abbiamo che:

$$T(s) = \frac{1GB - m}{b} = \frac{1GB - 6041585.56}{2.9 \pm 0.19} = 12 \pm 1 \text{ anni}$$

In cui, l'intervallo di confidenza al 95% è stato calcolato come segue:

$$CI = \frac{CI_{superiore} - CI_{inferiore}}{2} \approx 0.19$$

7. Reliability

In questo capitolo verranno condotte analisi di affidabilità su alcuni sistemi campione utilizzando tecniche di modellazione analitica. In parole semplici, l'affidabilità di un sistema indica la probabilità che esso funzioni correttamente per un periodo di tempo definito. Essa può essere influenzata da diversi fattori, tra cui la qualità dei componenti, la progettazione del sistema, la manutenzione e le condizioni ambientali. Le formule utilizzate durante l'analisi saranno le seguenti:

$$MTTF = \int_0^{\infty} R_{sys(t)} dt$$

Se $R_{sys} = e^{(-\lambda t)}$ allora:

$$MTTF = \int_0^{\infty} e^{-\lambda t} dt = \frac{1}{\lambda}$$

Se in un sistema i sottoinsiemi sono posti in :

❖ Serie: $R_{series}(t) = \prod_{i=1}^n R_i(t)$

❖ Parallelo: $R_{parallels}(t) = 1 - \prod_{i=1}^n (1 - R_i(t))$

Se in un sistema non si riesce ad individuare la serie e il parallelo si applica il **Teorema dell' Upper Bound**:

$$R_{sys} \leq UPP.BOUND$$

Dove:

$$UPP.BOUND = 1 - \prod_{i=1}^n (1 - R_{successpath_i}(t))$$

Ottobre è possibile usare il **condizionamento** dove si sceglie il sottosistema su cui si vuole condizionare (disabilitazione o meno del sottosistema). Si usa in questo caso il **Teorema della Probabilità Totale**:

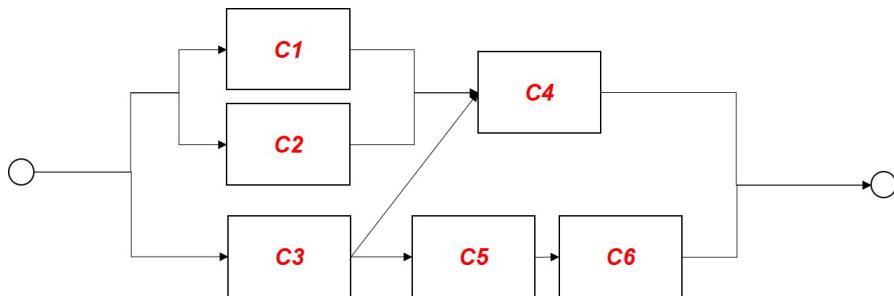
$$P(A) = \sum_i P(A|B_i)P(B_i)$$

Ovvero:

$$R_{sys} = R_m P(\text{system works}|m \text{ works}) + (1 - R_m)P(\text{system works}|m \text{ not works})$$

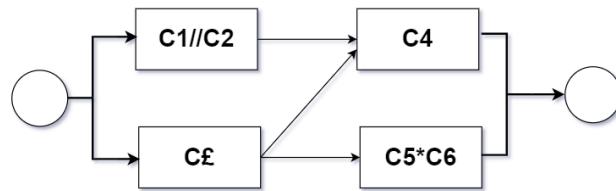
7.1 Esecizio 1

Sia dato il seguente schema:



$$R_1(t) = \dots = R_6(t) = R(t) = e^{-\lambda t}$$

Da qui si può notare che è possibile attuare delle semplificazioni, notando le serie e i paralleli.



$$R_{56} = R^2$$

$$R_{12} = 1 - (1 - R)^2$$

In questa forma il sistema non è né in serie né in parallelo. È necessario utilizzare il Teorema dell' Upper Bound e la tecnica del Conditioning.

Per applicare il Teorema dell' Upper Bound si enumerano i success path del sistema iniziale per poi posizionarli in parallelo.

$$C_1 \cdot C_4 = R^2$$

$$C_2 \cdot C_4 = R^2$$

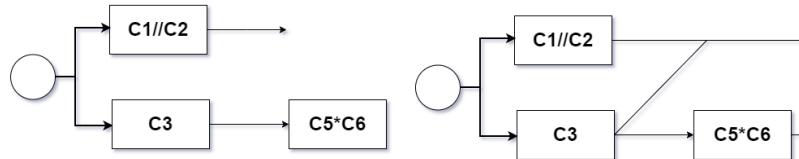
$$C_3 \cdot C_4 = R^2$$

$$C_3 \cdot C_5 \cdot C_6 = R^3$$

La R_{sys} avrà come Upper Bound il valore seguente:

$$R_{sys} \leq 1 - [(1 - R^2)^3(1 - R^3)] \leq -R^9 + 3R^7 + R^6 - 3R^5 - 3R^4 + R^3 + 3R^2$$

Si sta procedendo con il condizionamento del sistema in relazione al sottosistema C4. Questo processo permette di calcolare la reliability del sistema, considerando l'Upper Bound calcolato in precedenza. Il condizionamento offre la possibilità di analizzare il sistema nel suo complesso, sia in termini di funzionamento che di fallimento del sottosistema selezionato.



Viene applicata la probabilità totale per valutare R_{sys} :

$$R_{sys} = R_m P(sys \text{ works} | m \text{ works}) + (1 - R_m) P(sys \text{ works} | m \text{ fails})$$

Si nota che quando C4 lavora, [C1,C2] e C3 sono in parallelo, mentre quando C4 non lavora si ha che [C3,C5] e C6 sono in serie.

$$\begin{aligned} R_{sys} &= R \{1 - [1 - (1 - R)^2][1 - R]\} + (1 - R)R^3 \\ &= R \left[1 - (1 - (1 + R^2 - 2R))[1 - R]\right] + R^3 - R^4 \\ &= R[1 - (-R^2 + 2R)(1 - R)] + R^3 - R^4 \\ &= R(1 - (-R^2 + R^3 + 2R - 2R^2)) + R^3 - R^4 \\ &= R(1 + R^2 - R^3 - 2R + 2R^2) + R^3 - R^4 \\ &= R + R^3 - R^4 - 2R^2 + 2R^3 + R^3 - R^4 = -2R^4 + 4R^3 - 2R^2 + R \end{aligned}$$

Sapendo che $R = e^{-\lambda t}$ la Reliability complessiva del sistema è

$$R_{sys} = -2e^{-4\lambda t} + 4e^{-3\lambda t} - 2e^{-2\lambda t} + e^{-\lambda t}$$

Si nota che si tratta di un valore minore rispetto all'Upper Bound calcolato precedentemente. A questo punto è possibile calcolare l'MTTF del sistema

$$MTTF = \int_0^{+\infty} -2e^{-4\lambda t} + 4e^{-3\lambda t} - 2e^{-2\lambda t} + e^{-\lambda t} dt = -\frac{2}{4\lambda} + \frac{4}{3\lambda} - \frac{1}{\lambda} + \frac{1}{\lambda} = -\frac{1}{2\lambda} + \frac{4}{3\lambda} = \frac{5}{6\lambda}$$

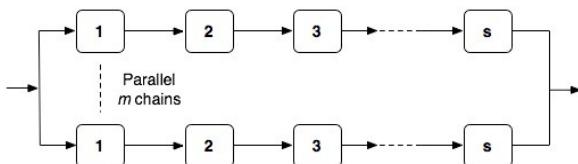
7.2 Esercizio 2

We want to compare two different schemes of increasing reliability of a system using redundancy. Suppose that the system needs s identical components in series for proper operation. Further suppose that we are given $(m \times s)$ components. Given that the reliability of an individual component is R , derive the expressions for the reliabilities of two configurations. For $m = 2$ and $s = 4$, compare the two expressions as function of a mission time t .

Let MTTF of the component be 800 hours.

Out of the two schemes shown in the figure below, which one will provide a higher reliability? Modify the scheme that has lower reliability in order to reach the same reliability of the other given the above MTTF (800 h).

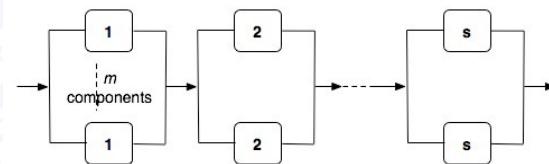
Il sistema si trova nella condizione in cui entrambe le configurazioni sono riconducibili ad una struttura serie-parallelo e di conseguenza è sufficiente calcolare la reliability e fare un confronto di quest'ultima assumendo $m=2$ ed $s=4$.



Il primo sistema presenta m successivi paths composti dalle m serie tra i componenti. Se fallisce anche un solo componente della catena, tutta la catena viene invalidata

$$\begin{aligned}
 R_{catena} &= R^s \\
 R_{sys1} &= 1 - (1 - R^s)^m \\
 R_{sys1} &= 1 - (1 - R^4)^2 = \\
 &= 2R^4 - R^8 \\
 MTTF &= \int_0^{+\infty} 2e^{-4\lambda t} - e^{-8\lambda t} dt = \\
 &= \frac{1}{2\lambda} - \frac{1}{8\lambda} \\
 800 &= \frac{3}{8\lambda} \\
 \lambda &= \frac{3}{8 \cdot 800} = 0.0004
 \end{aligned}$$

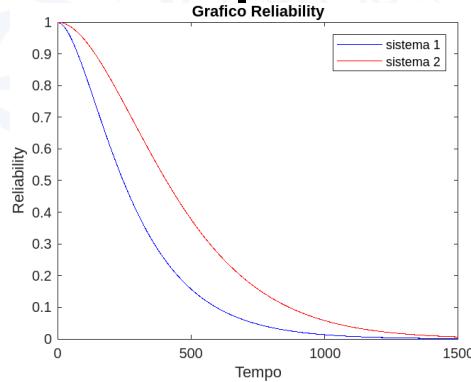
$$R_{sys1} = 2e^{-4\lambda t} - e^{-8\lambda t} = 0.47$$



Il secondo sistema presenta m^s successivi paths, il che ci suggerisce una reliability maggiore

$$\begin{aligned}
 R_{parallel} &= 1 - (1 - R)^m \\
 R_{sys2} &= (1 - (1 - R)^m)^s \\
 R_{sys2} &= (1 - (1 - R)^2)^4 = \\
 &= R^8 - 8R^7 + 24R^6 - 32R^5 + 16R^4 \\
 MTTF &= \int_0^{+\infty} e^{-8\lambda t} - 8e^{-7\lambda t} + 24e^{-6\lambda t} - 32e^{-5\lambda t} + 16e^{-4\lambda t} dt = \\
 &= \frac{1}{8\lambda} - \frac{8}{7\lambda} + \frac{8}{\lambda} - \frac{32}{5\lambda} \\
 800 &= \frac{163}{800\lambda} \\
 \lambda &= \frac{163}{280 \cdot 800} = 0.00072
 \end{aligned}$$

$$R_{sys2} = e^{-8\lambda t} - 8e^{-7\lambda t} + 24e^{-6\lambda t} - 32e^{-5\lambda t} + 16e^{-4\lambda t} = 0.42$$



Per ottenere una Reliability uguale è necessario aumentare quella del primo sistema. Si deve ottenere lo stesso numero di success path e quindi si devono andare ad egualare gli Upper Bound.

$$R_{sys1} = R_{sys2}$$

$$1 - (1 - R^s)^{m1} = (1 - (1 - R)^m)^s$$

$$1 - (1 - R^4)^{m1} = (1 - (1 - R)^2)^4$$

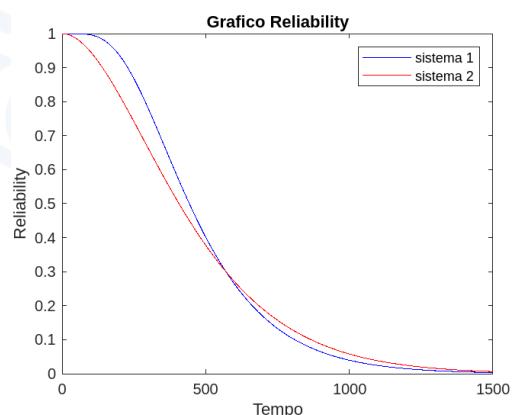
$$m1 = \frac{\ln\{(1 - (1 - R)^2)^4 - 1\} \cdot (-1)}{\ln(1 - R^4)}$$

Dove $\lambda = \frac{1}{MTTF}$; $R = e^{-\lambda t}$; $R(t)$ si può scegliere arbitrariamente ed $m_1 > 2$ per ottenere la stessa Reliability. Per il calcolo del valore effettivo di m_1 abbiamo considerato un mission time di 520h avendo un MTTF pari a 800.

$$R(520) = e^{-\frac{520}{800}} = 0.52$$

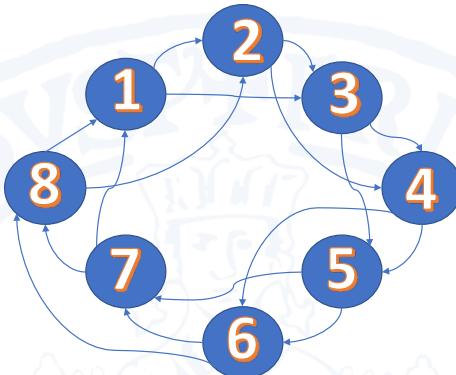
$$n = \frac{\ln\{(1 - (1 - 0.52)^2)^4 - 1\} \cdot (-1)}{\ln(1 - 0.52^4)} = 5.68 \cong 6$$

Il numero di success paths necessari ad avere uguale reliability per il sistema 1 è 6.



7.3 Esercizio 3

The architecture of a network of computers in a banking system is shown below. The architecture is called a skip-ring network and is designed to allow processors to communicate even after node failures have occurred. For example, if node 1 fails, node 8 can bypass the failed node by routing data over the alternative link connecting nodes 8 and 2. Assuming the links are perfect and the nodes each have a reliability of R_m , derive an expression for the reliability of the network. If R_m obeys the exponential failure law and the failure rate of each node is 0.005 failures per hour, determine the reliability of the system at the end of a 48-hour period.



$$R_{MN} = \sum_{i=0}^{N-M} \binom{N}{i} R_m^{N-i} (1-R_m)^i \rightarrow M - \text{out of } N \text{ systems}$$

Per il calcolo della Reliability per l'architettura di rete skip-ring vanno identificati tutti i possibili scenari di fallimento. Si può notare come non possono fallire più di 4 nodi.

$$R_{i=0} = \binom{8}{0} R_m^8 = R_m^8$$

Se si ha il fallimento di un solo nodo si avranno 8 differenti serie di componenti;

$$R_{i=1} = R_m^8 + \binom{8}{1} R_m^7 (1-R_m) = R_m^8 + \frac{8!}{1!(7!)} (1-R_m) = R_m^8 + 8R_m^7 (1-R_m)$$

Se si ha il fallimento di 2 nodi si avranno 20 combinazioni diverse di serie tra componenti;

$$\begin{aligned} R_{i=2} &= R_m^8 + 8R_m^7 (1-R_m) + \left[\binom{8}{2} - 8 \right] R_m^6 (1-R_m)^2 = \\ &= R_m^8 + 8R_m^7 (1-R_m) + R_m^6 \left[\frac{8!}{2! \cdot 6!} - 8 \right] (1-R_m)^2 = R_m^8 + 8R_m^7 (1-R_m) + 20R_m^6 (1-R_m)^2 \end{aligned}$$

Se si ha il fallimento di 3 nodi si avranno 16 possibili serie di componenti;

$$R_{i=3} = R_m^8 + 8R_m^7(1 - R_m) + 20R_m^6(1 - R_m)^2 + \left[\binom{8}{3} - 32 * 8 \right] R_m^5(1 - R_m)^3 = \\ = R_m^8 + 8R_m^7(1 - R_m) + 20R_m^6(1 - R_m)^2 + 16R_m^5(1 - R_m)^3$$

Se si ha il fallimento di 4 nodi si hanno solo 2 combinazioni possibili di serie tra componenti.

$$R_{i=4} = R_m^8 + 8R_m^7(1 - R_m) + 20R_m^6(1 - R_m)^2 + 16R_m^5(1 - R_m)^3 \left[\binom{8}{4} - 32 * 8 * 28 \right] R_m^4(1 - R_m)^4 = \\ = R_m^8 + 8R_m^7(1 - R_m) + 20R_m^6(1 - R_m)^2 + 16R_m^5(1 - R_m)^3 + 2R_m^4(1 - R_m)^4$$

Volendo calcolare la Reliability per un mission time pari a 48 sostituiamo nell'espressione della reliability

$\lambda = 0.005$ e $t=48$ ed otteniamo:

$$R(48) = e^{-0.005 \cdot 48} = 0.786 \\ R_{sys} = 0.727$$

7.4 Esercizio 4

Compare the reliability of the following schemes, assuming an exponential failure occurrence with following values:

$$\text{MTTFA} = 500 \text{ h} \rightarrow R_A(t) = e^{-t/500} = e^{-0.02t}$$

$$\text{MTTFB} = 9000 \text{ h} \rightarrow R_B(t) = e^{-t/9000} = e^{-0.0001t}$$

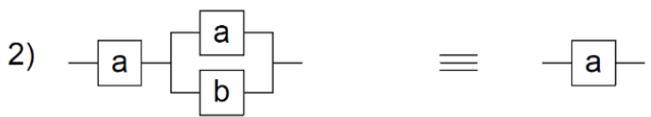
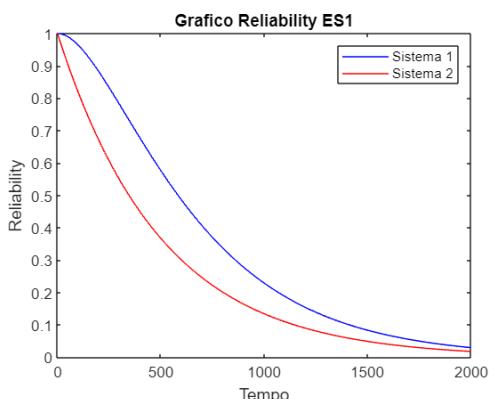
$$\text{MMTFC} = 10000 \text{ h} \rightarrow R_C(t) = e^{-t/10000} = e^{-0.0001t}$$



$$R_{sys1} = 1 - (1 - R_A R_B)(1 - R_A R_C)$$

$$R_{sys2} = R_A [1 - (1 - R_B)(1 - R_C)]$$

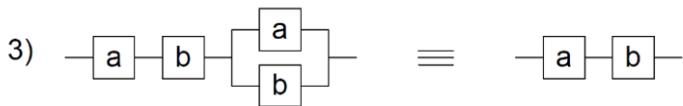
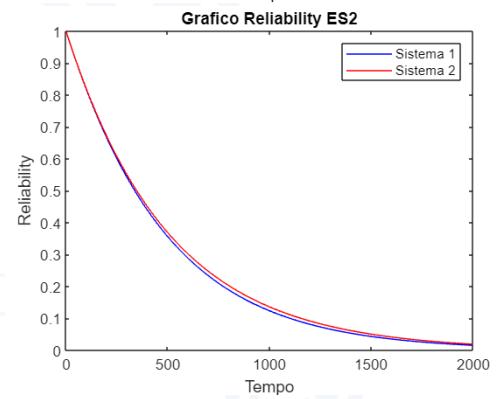
$$R_{sys1} > R_{sys2}$$



$$R_{sys1} = R_A [1 - (1 - R_B)(1 - R_A)]$$

$$R_{sys2} = R_A$$

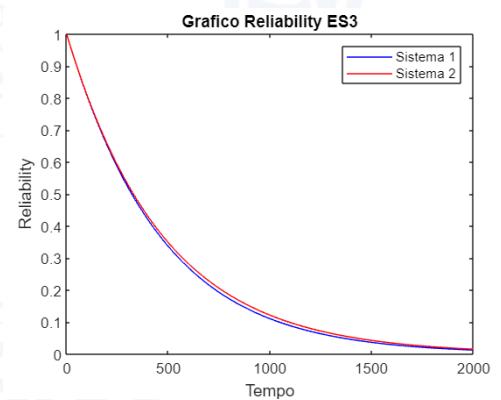
$$R_{sys1} < R_{sys2}$$



$$R_{sys1} = R_A R_B [1 - (1 - R_A)(1 - R_B)]$$

$$R_{sys2} = R_A + R_B$$

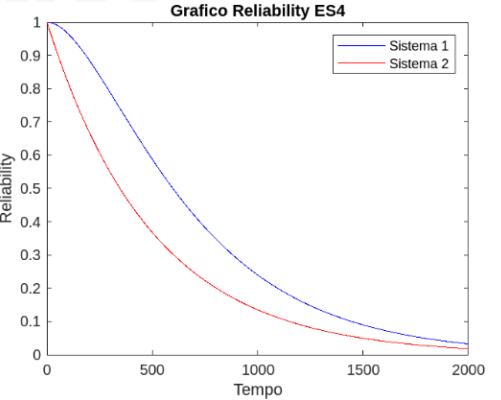
$$R_{sys1} < R_{sys2}$$



$$R_{sys1} = [1 - (1 - R_A)(1 - R_A R_B)]$$

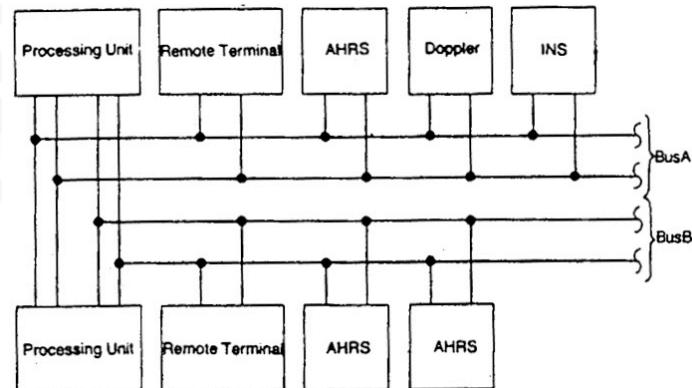
$$R_{sys2} = R_A$$

$$R_{sys1} > R_{sys2}$$



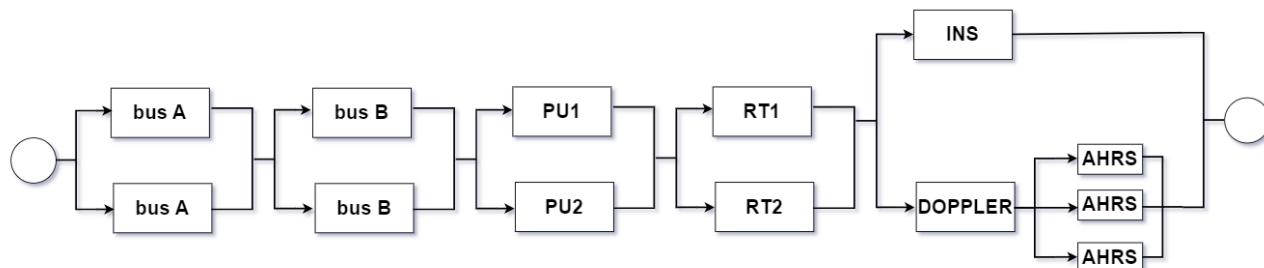
7.5 Esercizio 5

The system shown in the figure below is a processing system for a helicopter. The system has dual-redundant processors and dual-redundant interface units. Two buses are used in the system, and each bus is also dual-redundant. The interesting part of the system is the navigation equipment. The aircraft can be completely navigated using the Inertial Navigation System (INS). If the INS fails, the aircraft can be navigated using the combination of the Doppler and the altitude heading and reference system (AHRS). The system contains three AHRS units, of which only one is needed. This is an example of functional redundancy where the data from the AHRS and the Doppler can be used to replace the INS, if the INS fails. Because of the other sensors and instrumentation, both buses are required for the system to function properly regardless of which navigation mode is being employed.

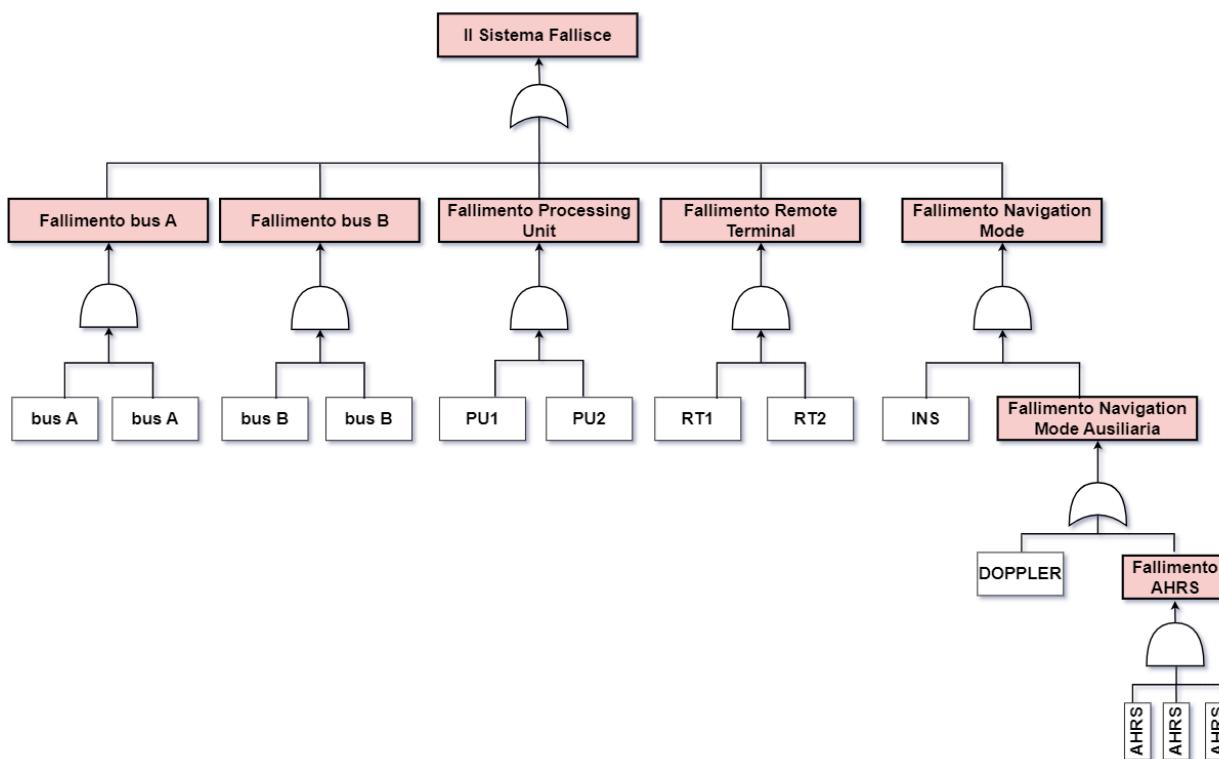


- Draw the reliability block diagram of the system.
- Draw the Fault Tree of the system and analyze the minimal cutsets.
- Calculate the reliability for a one-hour flight using the MTTF figures given in the table below. Assume that the exponential failure law applies and that the fault coverage is perfect.
- Repeat (c), but this time, incorporate a coverage factor for the fault detection and reconfiguration of the processing units. Using the same failure data, determine the approximate fault coverage value that is required to obtain a reliability (at the end of one hour) of 0.99999.

- a) Prima di procedere con la realizzazione del Diagramma a Blocchi di Affidabilità (RBD), sono state effettuate le seguenti considerazioni:
- ❖ È presente una configurazione dual-redundant per i BUS, le unità di elaborazione (Processing Unit) e i terminali remoti (Remote Terminal).
 - ❖ I blocchi simili sono disposti in parallelo.
 - ❖ In serie si trova il parallelo tra il componente INS e la serie del componente Doppler con il parallelo dei 3 componenti AHRS.
 - ❖ In base a queste considerazioni, è stato realizzato il seguente RBD:



- b) A questo punto è facile passare alla realizzazione del Fault Tree mettendo in OR i componenti in serie ed in AND i componenti in parallelo:



Per evidenziare i minimal cut-set del sistema, è utile esprimere l'albero dei guasti (Fault Tree) in forma analitica tramite la sua funzione. Tale funzione associa ad ogni AND il prodotto tra i componenti coinvolti nell' operazione, mentre alle OR vengono associate le somme tra i componenti coinvolti nell' operazione.

Otteniamo:

$$\phi = (bus_A \cdot bus_A) + (bus_B \cdot bus_B) + (PU \cdot PU) + (RT \cdot RT) + (INS \cdot Doppler) + (AHRS^3 \cdot INS)$$

È semplice, a questo punto, determinare i minimal cut-set sulla base dei mintermini presenti all'interno della funzione e sono i seguenti:

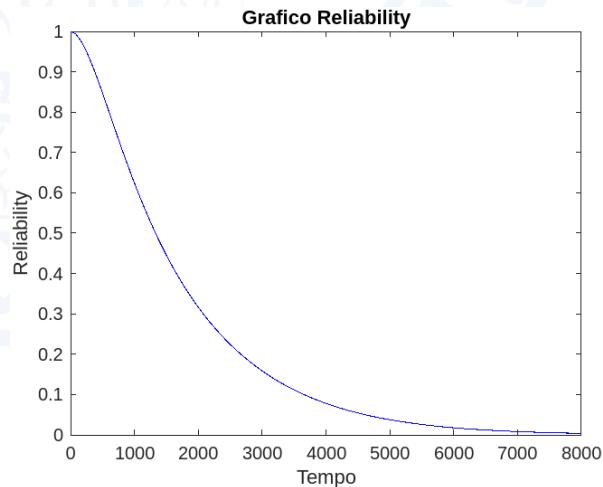
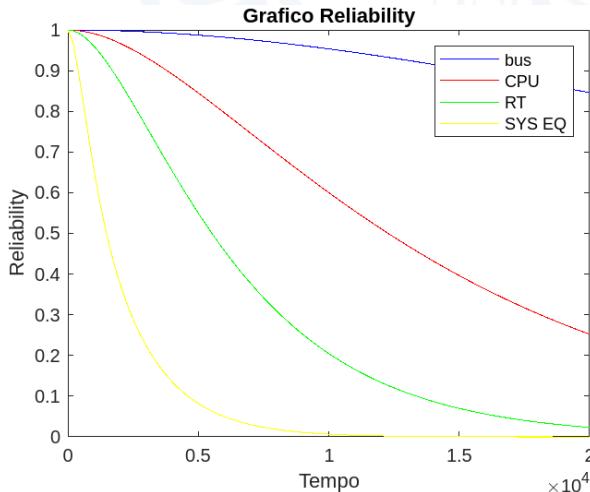
$$\begin{aligned} MCS_1 &= \{bus_A \cdot bus_A\} \\ MCS_2 &= \{bus_B \cdot bus_B\} \\ MCS_3 &= \{PU \cdot PU\} \\ MCS_4 &= \{RT \cdot RT\} \\ MCS_5 &= \{INS \cdot Doppler\} \\ MCS_6 &= \{AHRS^3 \cdot INS\} \end{aligned}$$

- c) Per il calcolo della Reliability del sistema possiamo procedere risolvendo il Reliability Block Diagram.

Component	MTTF
BUS	60000
PU	10000
RT	4500
INS	2000
DOPPLER	500
AHRS	2000

$$R_{sys} = [1 - (1 - R_{BUS})^2]^2 [1 - (1 - R_{PU})^2] [1 - (1 - R_{RT})^2] [1 - (1 - R_{INS})(1 - (1 - R_{AHRS})^3)]$$

$$\text{Con } R = e^{-\lambda t}, \frac{1}{\lambda} = MTTF$$



La

Reliability al tempo $t=1$ dopo 1h di volo è:

$$R_{sys}(1) = 0.999998941322750356519$$

- d) Fino ad ora si è supposto che ogni sistema sia in grado di rilevare perfettamente un fallimento. Considerando che un componente possa non riconoscere il proprio fallimento conseguentemente il sistema riadattarsi è importante considerare il fattore di coverage. Dal grafico precedente sulla Reliability mostrata per le singole componenti si è scelto di considerare il fattore di coverage per il componente di Processing Unit. La Reliability del parallelo dei componenti di Processing Unit è data dalla probabilità che una replica funzioni più la probabilità che questa stessa fallisca per la probabilità di rilevare il fallimento, moltiplicata la probabilità che l'altra replica funzioni.

$$[1 - (1 - R_{PU})^2] = [R_{PU} + c(1 - R_{PU})R_{PU}]$$

Per ottenere una reliability maggiore di 0.99999 dopo 1 ora è sufficiente calcolare R_{sys} andando ad effettuare la sostituzione del parallelo relativo al componente di Processing Unit e risolvere la disequazione nella sola variabile c.

$$R_{sys[c]} \geq 0.99999$$

$$[1 - (1 - R_{BUS})^2]^2 [R_{PU} + c(1 - R_{PU})R_{PU}] [1 - (1 - R_{RT})^2][1 - (1 - R_{INS})(1 - [R_D(1 - (1 - R_{AHRS})^3)])] \geq 0.99999$$

L'esercizio è stato risolto in Matlab ed abbiamo ottenuto un valore di C che deve essere maggiore o uguale di 0.910573.

8. Field Failure Data Analysis

La **Field Failure Data Analysis** è un processo tecnico che coinvolge l'analisi dei dati provenienti da guasti o malfunzionamenti riscontrati. L'obiettivo principale è identificare le cause sottostanti dei fallimenti per migliorare la progettazione, la produzione o altri aspetti del ciclo di vita del prodotto, riducendo così i rischi di guasto futuro.

La tecnica dell'FFDA è composta da diverse fasi:

- ❖ **Logging Collection:** Questa fase coinvolge la raccolta e la memorizzazione dei dati generati dal sistema durante il suo funzionamento. Affinché il Logger svolga un monitoraggio mirato, è essenziale configurarlo adeguatamente. Il file risultante contiene **dati grezzi**.
- ❖ **Filtering:** In questa fase, si procede a un'operazione di filtraggio dei dati raccolti al fine di concentrare l'analisi solo su quelli rilevanti. Questo processo avviene in base al log di interesse, utilizzando approcci come il White Listing o il Black Listing. Il file ottenuto è quindi **filtrato** e pronto per ulteriori manipolazioni.
- ❖ **Data Manipulation:** I dati vengono aggregati attraverso un processo di coalescenza. Questa operazione consente di raggruppare i fallimenti relativi allo stesso guasto all'interno di una singola entità, detta **tupla**, riducendo in modo significativo le dimensioni del dataset iniziale.
- ❖ **Analysis:** In questa fase, i dati subiscono un'analisi per vari scopi. Ciò può comprendere la determinazione della curva di affidabilità del sistema o l'identificazione della distribuzione del tempo fino al guasto. L'obiettivo è estrapolare informazioni cruciali che possono spaziare dalla salute del sistema all'anticipazione dei tempi di guasto.

Nel corso del capitolo andremo ad effettuare delle analisi a partire dai log di due supercalcolatori **Mercury** e **Blue Gene**.



8.2 Mercury

Il supercomputer Mercury, sviluppato da IBM, vanta una potenza computazionale con ben 1774 processori e una memoria di 3 GB per ogni nodo. Attualmente collocato presso il National Center for Supercomputing Applications presso l'Università di Illinois a Urbana-Champaign, Mercury si attesta al 15º posto nella classifica dei supercomputer più veloci al mondo, secondo il sito Top-500 Supercomputers, datato giugno 2004.

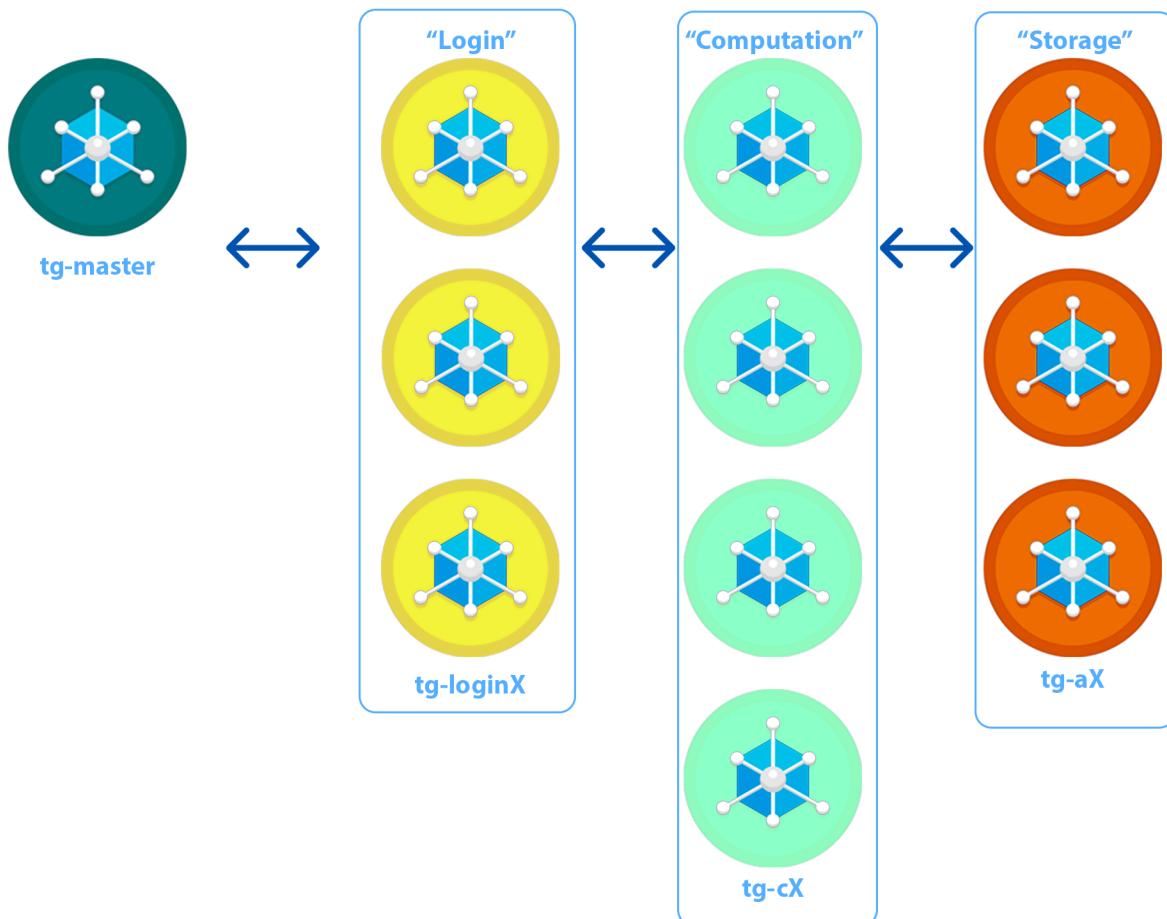


Figura 8.1: Schema del supercalcolatore Mercury

È possibile identificare quattro tipologie di nodi all'interno dell'ambiente:

- ❖ **tg-master:** Nodo adibito alla gestione e al controllo del sistema.;
- ❖ **tg-loginX:** Nodi dedicati all'esecuzione dei servizi finalizzati a garantire un accesso adeguato alle risorse di sistema;
- ❖ **tg-cX:** Nodi designati per l'esecuzione di processi intensivi dal punto di vista computazionale.
- ❖ **tg-sX:** Nodi destinati alla memorizzazione dei dati, dove avviene l'archiviazione e la gestione delle informazioni.

8.2.2 Logging collection e Filtering

Questa fase è stata eseguita già precedentemente e ha comportato la raccolta di dati archiviati nel file "MercuryErrorLog.txt", il quale contiene 80854 voci relative a errori fatali del sistema. I registri sono stati già sottoposti a de-parametrizzazione e formattati secondo la seguente struttura:

- ❖ timestamp
- ❖ nodo che ha dato origine al failure
- ❖ categoria di failure
- ❖ messaggio di failure

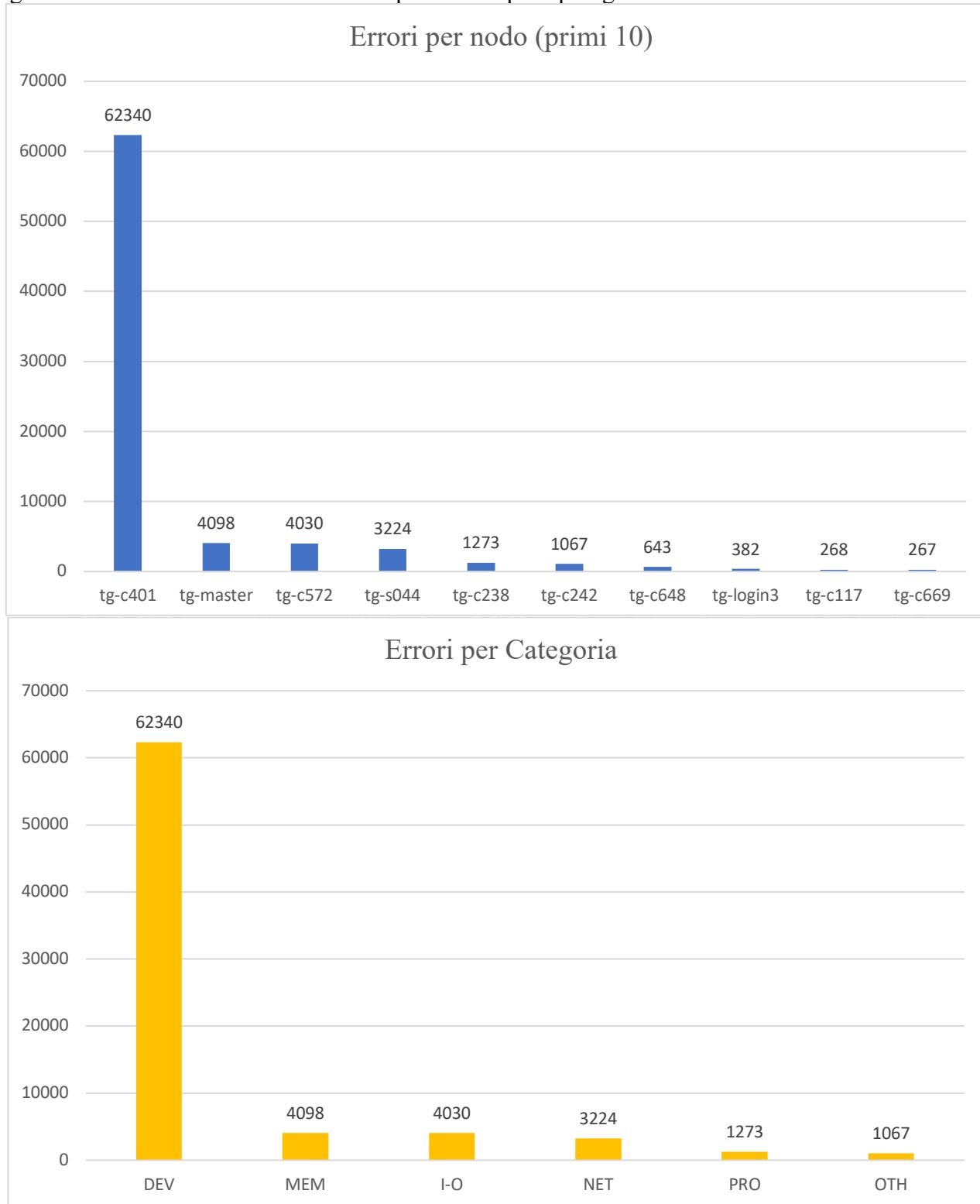
un esempio di alcune entry del file:

```
1. 1167657137 tg-c238 DEV +BEGIN HARDWARE ERROR STATE AT CPE
2. 1167659592 tg-c238 MEM Physical Address x, Address Mask: x, Node: x, Card: x, Module: x, Bank: x, Device: x,
Row: x, Column: x,
```

le categorie di failure riscontrate sono:

- ❖ DEV: Errori relativi alla sezione di informazioni sul componente di errore della piattaforma PCI.
- ❖ MEM: Errori di Memoria come Physical Address, Address Mask, Node, Card, Module, Bank, Device, Row e Column
- ❖ NET: Connection down
- ❖ I-O: errori di input e output
- ❖ PRO: begin hardware error

Di seguito sono mostrati il numero di errori per nodo e per tipologia di errore.



8.2.3 Data manipulation

Nella fase di manipulation, si procede all'identificazione dei fallimenti del sistema basandosi sul registro di sistema. Per individuare tali fallimenti, viene adottata la tecnica della Coalescenza, la quale è in grado di rilevare la correlazione tra le voci presenti nel file di log. Questa metodologia consente di ridurre l'insieme delle informazioni raccolte durante la fase di logging collection.

Attraverso operazioni di aggregazione o eliminazione, la Coalescenza consente di rimuovere dati ridondanti o equivalenti tra di essi. Tale procedura è necessaria in quanto un guasto può generare diversi fallimenti nel sistema, e poiché gli effetti di un guasto si propagano attraverso il sistema, vengono rilevati più volte da componenti diverse. Pertanto, è opportuno raggruppare tutti i fallimenti correlati allo stesso guasto in un singolo evento, riducendo così l'elenco dei log.

In particolare, si utilizza la tecnica della coalescenza temporale, la cui valutazione è basata sulla seguente condizione:

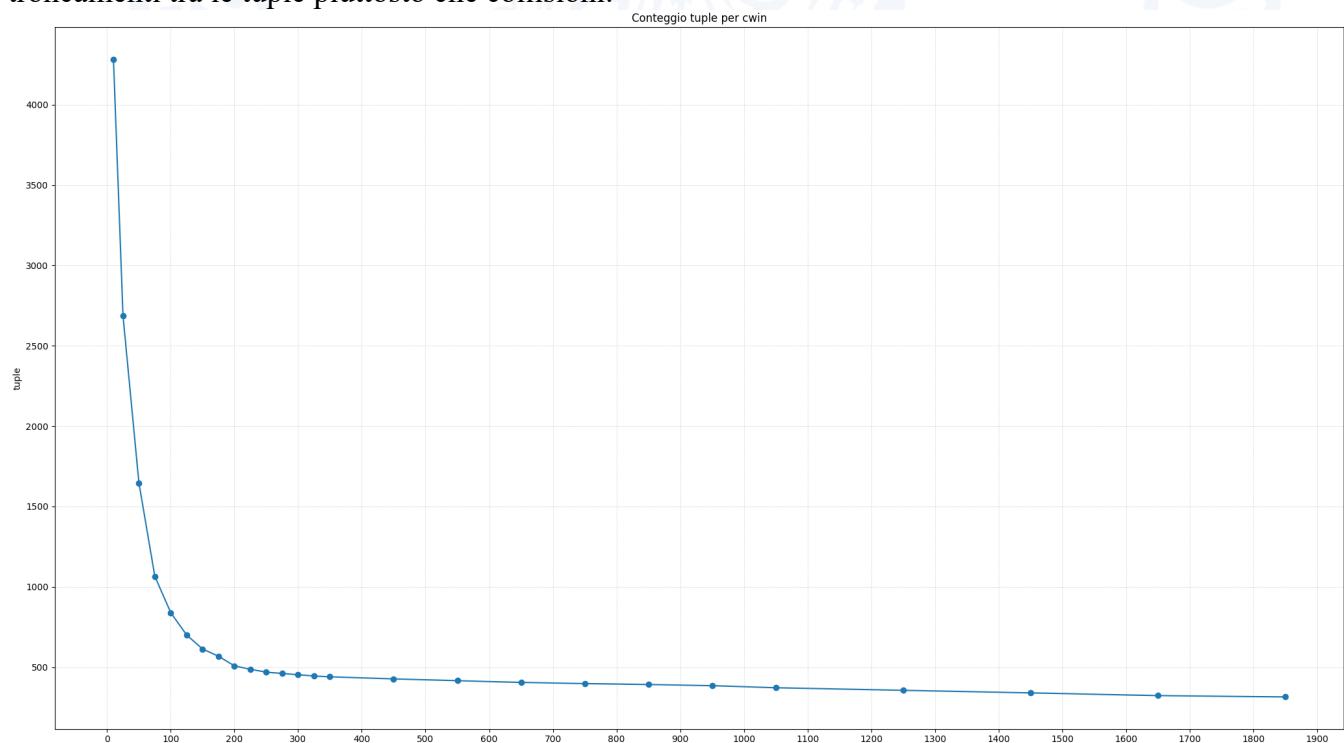
$$\text{IF } t(X_{i+1}) - t(X_i) < W \text{ THEN ADD } X_{i+1} \text{ to the tuple}$$

Dove:

- ❖ X_i è la i -esima entry nel log;
- ❖ $t(X_i)$ è il timestamp della X_i entry;
- ❖ W è la Coalescence Window (CWIN).

In sintesi, questa condizione stabilisce che se la differenza temporale tra due voci consecutive è entro il valore specificato dalla finestra temporale di coalescenza (CWIN), allora queste voci sono considerate correlate e vengono aggregate insieme.

La determinazione della dimensione della finestra di coalescenza riveste un ruolo cruciale in quanto, se si adotta una finestra troppo ampia, possono verificarsi collisioni. Le collisioni si manifestano quando fallimenti distinti sono inclusi all'interno di una stessa tupla tra le voci del registro. In alternativa, l'adozione di una finestra di coalescenza eccessivamente ridotta può condurre a troncamenti, ossia all'inserimento di eventi relativi allo stesso fallimento in tuple differenti. Si cerca dunque di preferire una Coalescence Window piccola al fine di effettuare una stima pessimistica della curva di reliability. Si accettano quindi più volentieri troncamenti tra le tuple piuttosto che collisioni.



Dunque si è scelto un CWIN pari a 200s, punto con salto maggiore, che presenta 508 tuple.

Successivamente alla determinazione della finestra temporale migliore, è stato eseguito uno script in linguaggio Python denominato "UStupling with CWIN.py". Tale script genera un file distintivo per ciascuna entry, fornendo ulteriori dettagli, tra cui:

- ❖ **Lunghezza**, definita come la differenza tra i timestamp della prima e dell'ultima entry all'interno di ciascuna tupla, rappresentando così la durata complessiva della tupla espressa in secondi.

$$length_i = timestamp_{N,i} - timestamp_{1,i}$$

dove N è l'ultima entry della tupla ed i è la tupla i-esima.

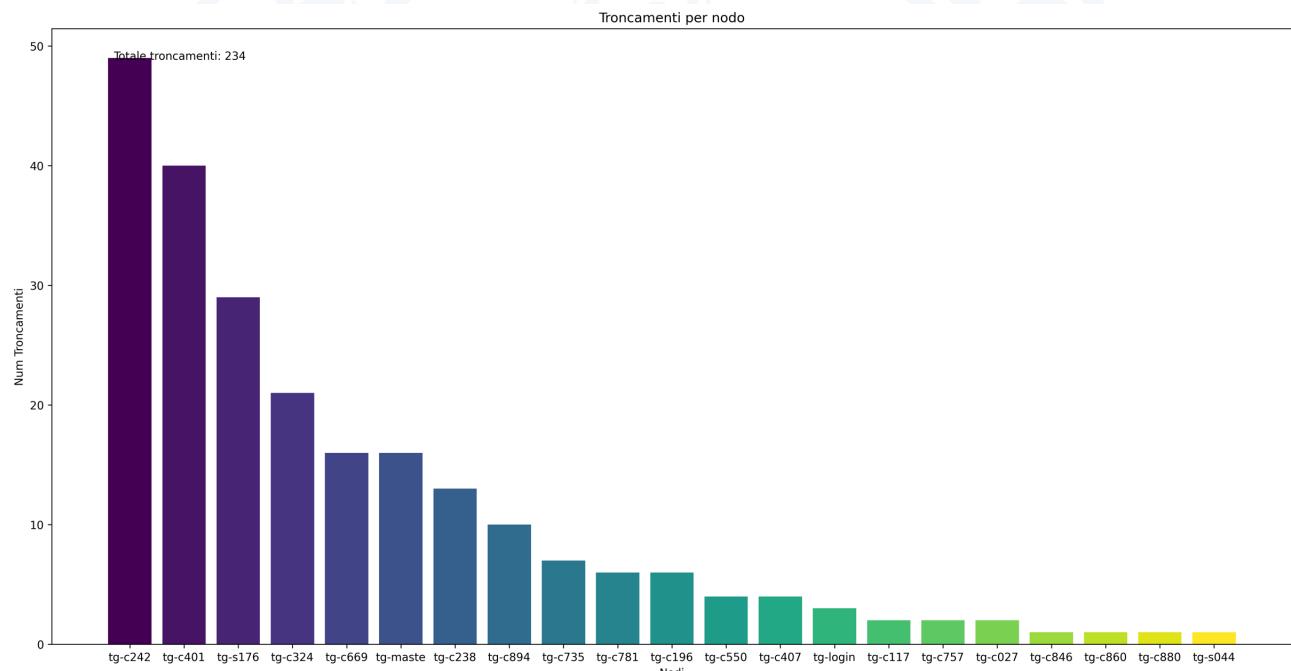
- ❖ **Punto di inizio**, timestamp della prima entry della tupla con annessa durata della stessa;

- ❖ **Interarrivo**, calcolato come il tempo in secondi trascorso tra la prima entry della tupla corrente e l'ultima voce della tupla precedente (TTF - Time To Failure).

$$interarrivo_s = timestamp_{1,j} - timestamp_{N,i-1}$$

dove N rappresenta l'indice dell'ultima voce della tupla, i denota la tupla corrente e s è il numero dell'interarrivo s-esimo (con $2 \leq i \leq s$).

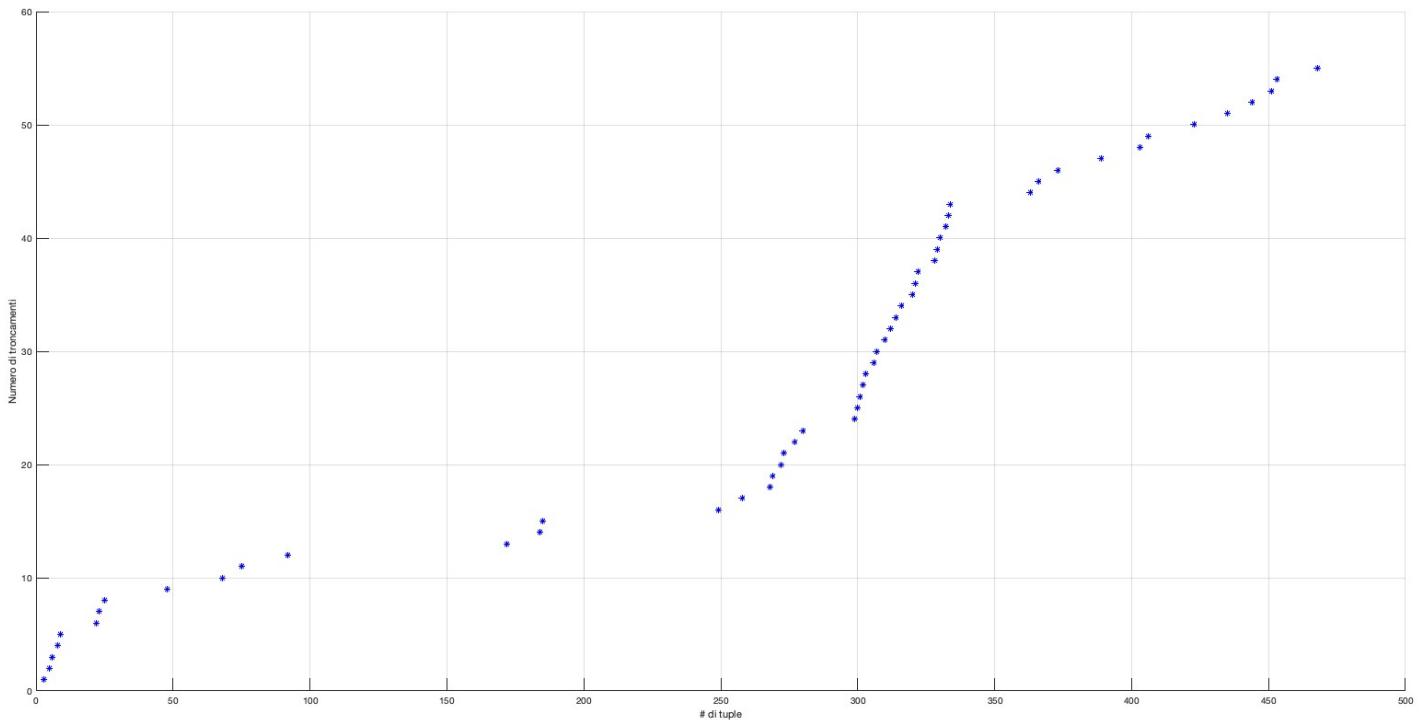
Tramite lo script python *TroncamentiCollisioni.py* si è andata a calcolare una ipotetica stima dei possibili troncamenti e collisioni.



Sono stati riscontrati un totale di 234 collisioni generati però nella maggioranza dai nodi tg-c242, tg-c401, tg-s176 e tg-c324.

Le collisioni riscontrate dallo script sono invece 47.

Utilizzando lo script Matlab troncamenti.m invece, si è considerato gli interarrivi brevi, nel nostro caso minori della finestra della finestra temporale + 50%.

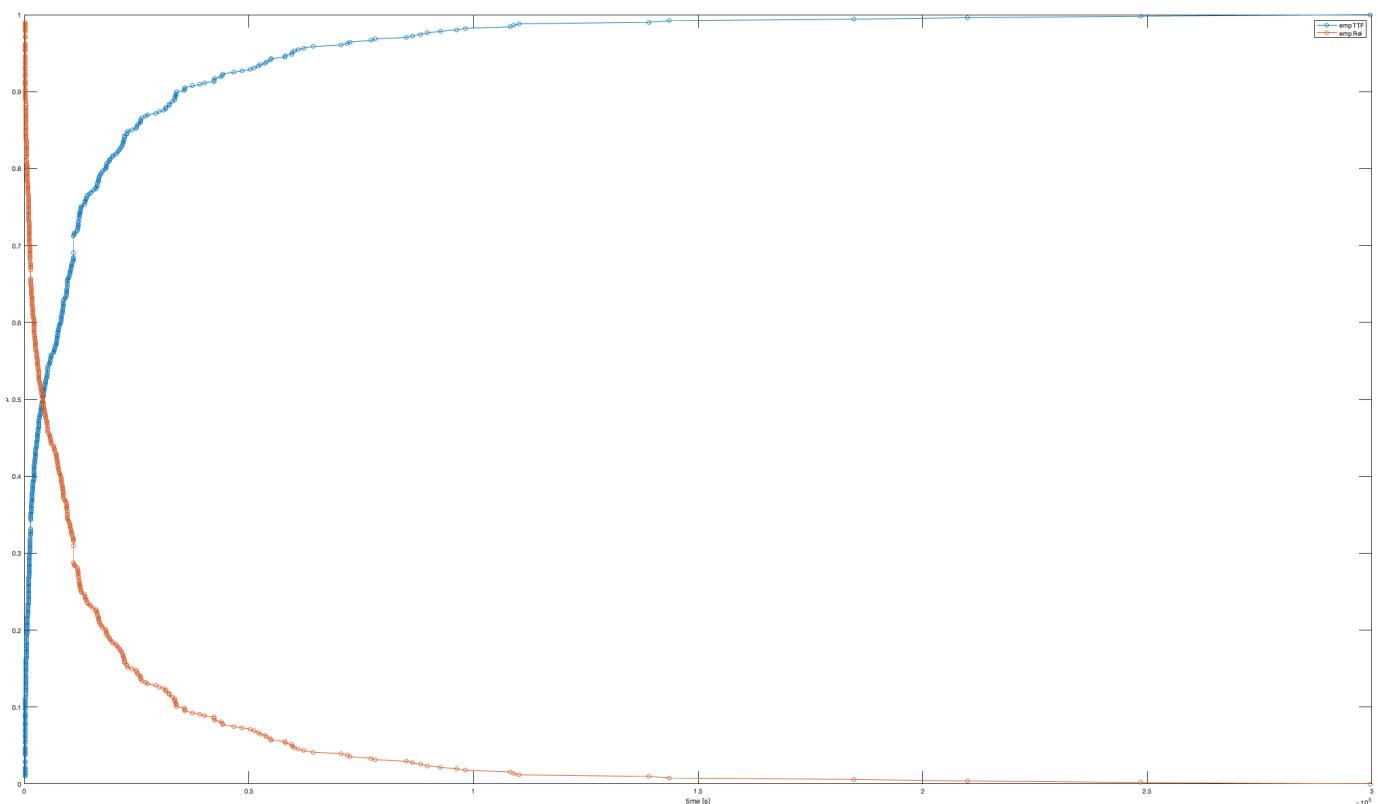


Da questo grafico è possibile verificare che la maggior parte dei troncamenti sarà verosimilmente collocata tra la 300 e 350.

I risultati ottenuti sembrano accettabili e di conseguenza si ritiene 200s un buon valore di grandezza della finestra di coalescenza.

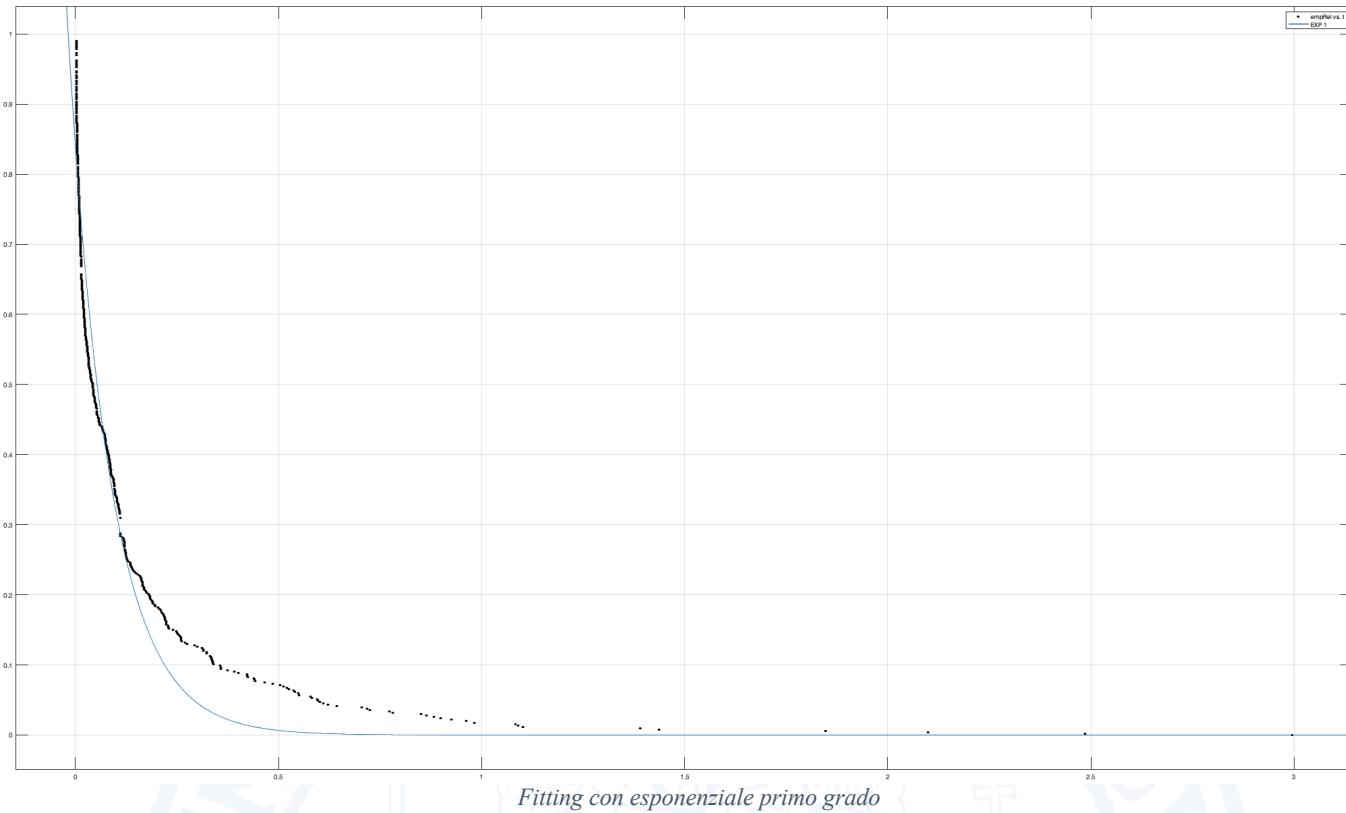
8.2.4 Data Analysis

In questa fase, viene condotto uno studio sulla affidabilità del sistema basato sui risultati ottenuti in fasi precedenti. Attraverso il calcolo degli intervalli di tempo tra gli eventi (interarrivi), è possibile estrarre dalla raccolta dati la distribuzione empirica della probabilità del tempo fino al guasto (TTF) del sistema, utilizzando uno script in MATLAB. Di conseguenza, il calcolo della Reliability del sistema, indicata come R, viene eseguito mediante la formula $R = 1 - TTF$.

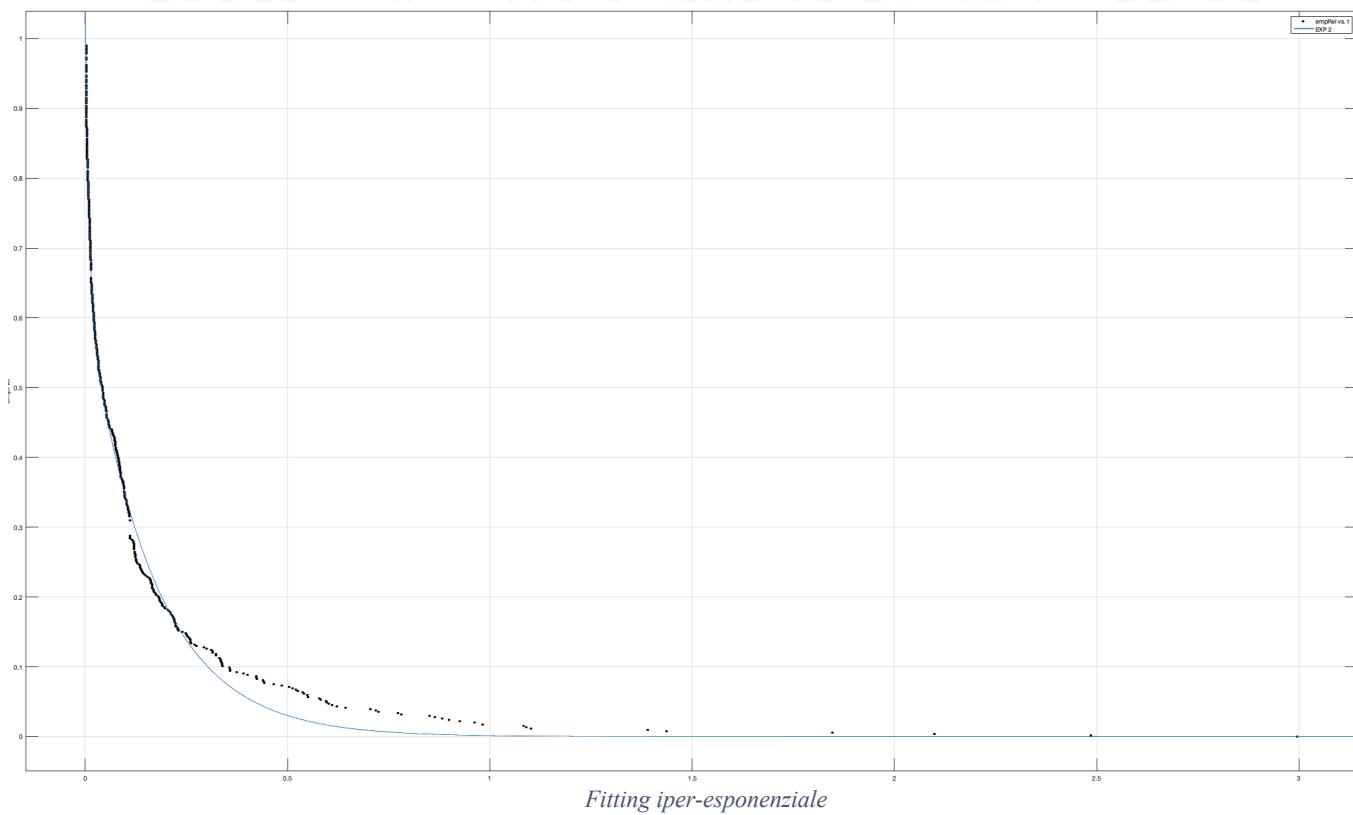


Confronto tra la Reliability empirica e Unreliability empirica - Mercury

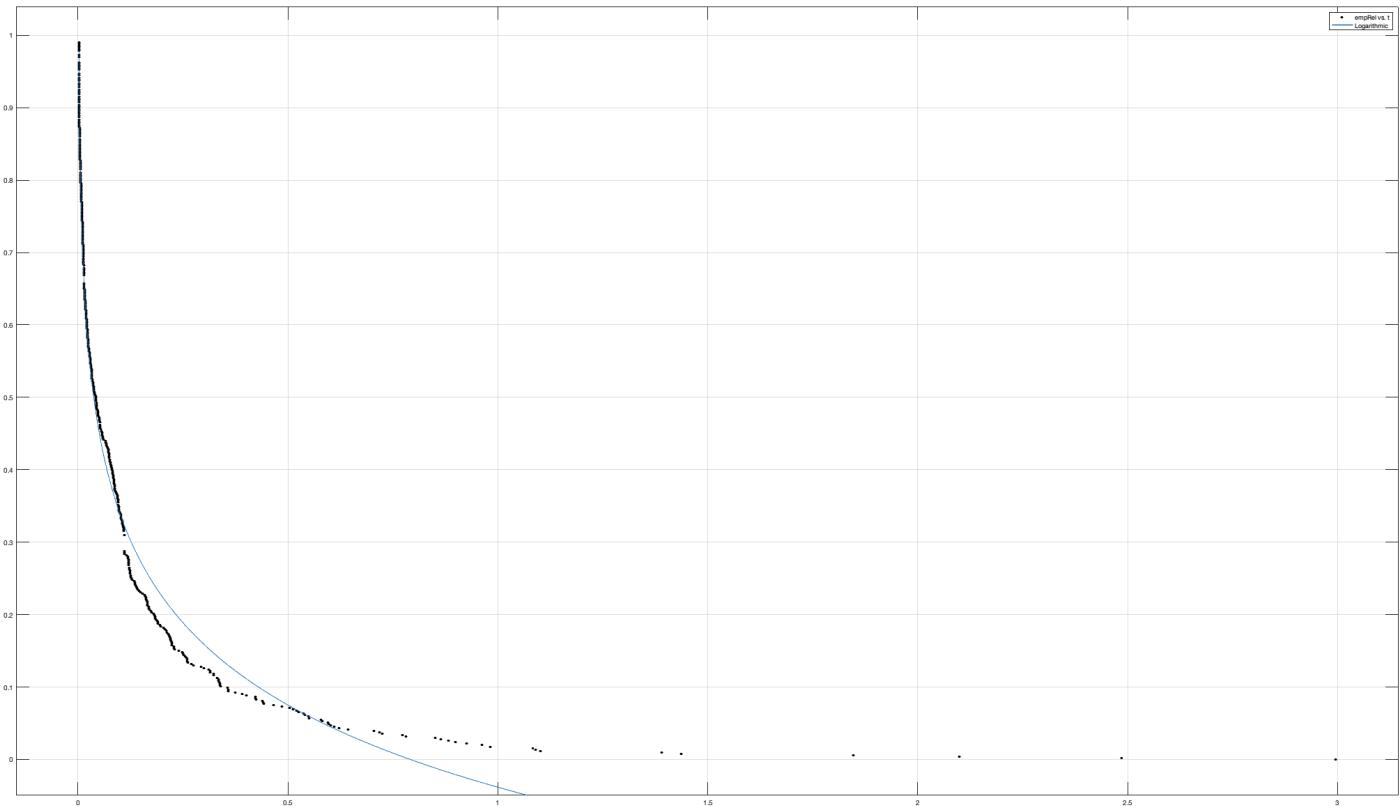
Dato un tempo t , la funzione $TTF(t)$ rappresenta la probabilità che il sistema manifesti un guasto dopo un periodo di t secondi dall'ultima operazione di ripristino. Attraverso l'uso dello strumento `cftool`, incluso nella toolbox di statistica di MATLAB, è possibile derivare una distribuzione teorica basata sulla distribuzione empirica. Questo processo coinvolge il tentativo di modellare l'affidabilità del sistema utilizzando una distribuzione teorica ben definita. Per trovare la distribuzione teorica che meglio approssima quella empirica, si procede per tentativi, confrontando la distribuzione empirica con diverse distribuzioni ben note della teoria delle probabilità, come la distribuzione esponenziale, Weibull, e altre.



Fitting con esponenziale primo grado



Fitting iper-esponenziale



8.2 Fitting con logaritmica

Di seguito sono presentati i parametri di qualità del fitting:

Fit type	R-square	SSE	DFE	Adj R-sq	RMSE	# Coeff
exp1	0.95343	1.7471	464	0.95333	0.061362	2
exp2	0.99612	0.14572	462	0.99609	0.01776	4
log	0.98806	0.44811	464	0.98803	0.031076	2

Dalle analisi visive sembrerebbe migliore la distribuzione iper-esponenziale.

Come ulteriore verifica si effettua un *Kolmogorov-Smirnov test* utilizzando il seguente comando su matlab:

```
1. [h,p,k] = kstest2(empRel,fittedmodel(t))
```

dove:

- ❖ H: Uguale a 1 se rigetta l'ipotesi nulla al 5% di livello di significatività, 0 altrimenti. Di conseguenza se è 0, con un livello di confidenza del 95% le distribuzioni provengono dalla stessa distribuzione;
- ❖ P: Quanto più alto è il valore del p-value, maggiore è la probabilità che l'ipotesi nulla non venga rigettata
- ❖ K: valore del test definito come $K = [\max(EmpRel - fittedRel)]$

Vengono mostrati di seguito i risultati ottenuti:

	h	p	k
Esponenziale	1	3.4090e-206	1
Iper-esponenziale	0	0.6666	0.0472
Logaritmica	0	0.4521	0.0558

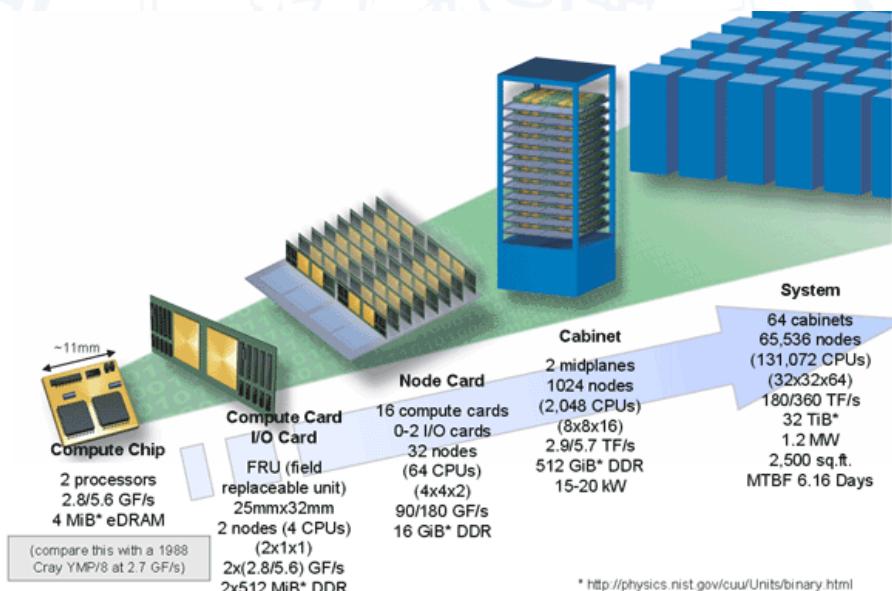
Tramite il test si è verificato quindi che la distribuzione iper-esponenziale è il modello migliore.

8.3 Blue Gene/L



Il progetto Blue Gene, sviluppato da IBM, è stato concepito con l'obiettivo di progettare supercomputer in grado di raggiungere velocità operative nell'ordine dei petaFLOPS mantenendo al contempo un basso consumo energetico. Il progetto Blue Gene ha dato vita a tre generazioni di supercomputer: Blue Gene/L, Blue Gene/P e Blue Gene/Q. Durante la loro implementazione, i sistemi Blue Gene hanno costantemente dominato le classifiche TOP500 e Green500, rispettivamente per i supercomputer più potenti e più efficienti dal punto di vista energetico.

Il Calcolatore Blue Gene presenta un'architettura complessa che include diversi componenti chiave, ciascuno con il suo ruolo specifico. Questi componenti sono organizzati in un'architettura gerarchica che comprende rack(RX), Nodi(NX), schede di calcolo(JX) e schede di I/O(J18) e Compute Chip(I01,U11).



Le dimensioni del sistema sono maggiori del sistema Mercury analizzato precedentemente.

8.3.1 Logging collection e Filtering

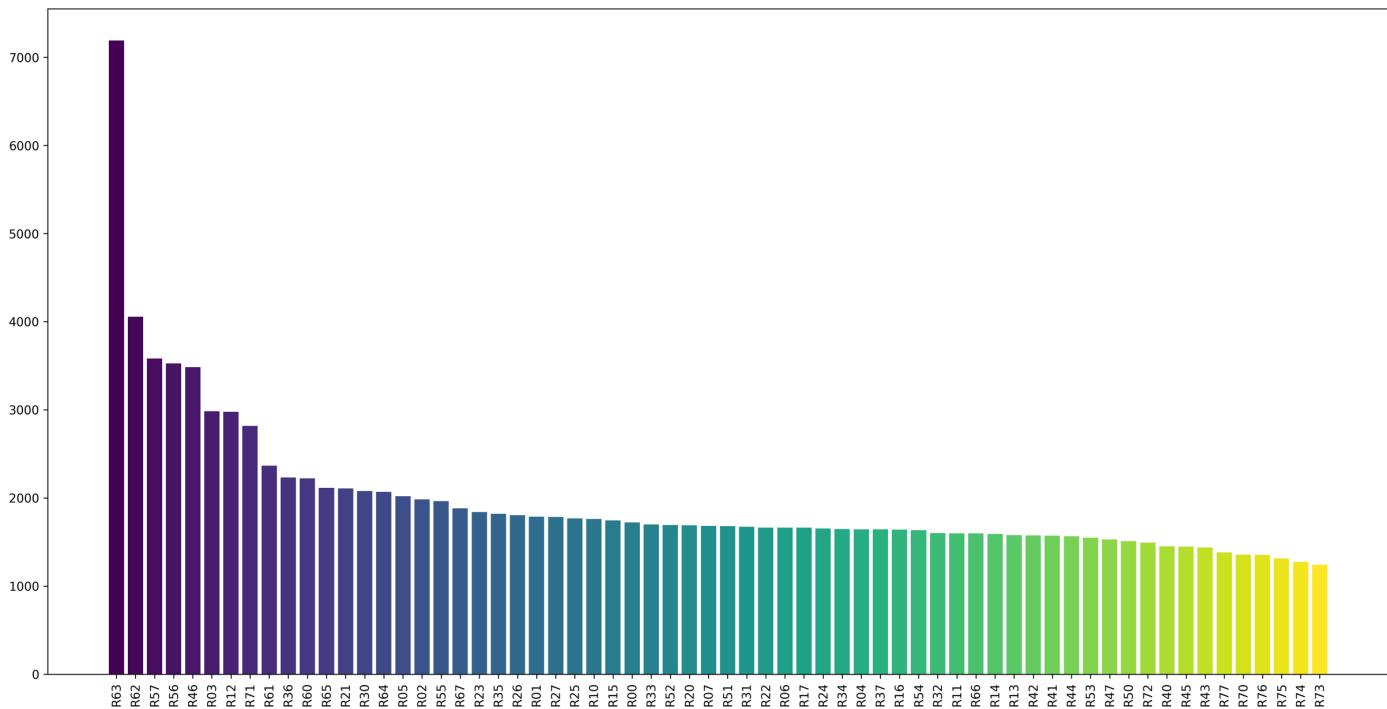
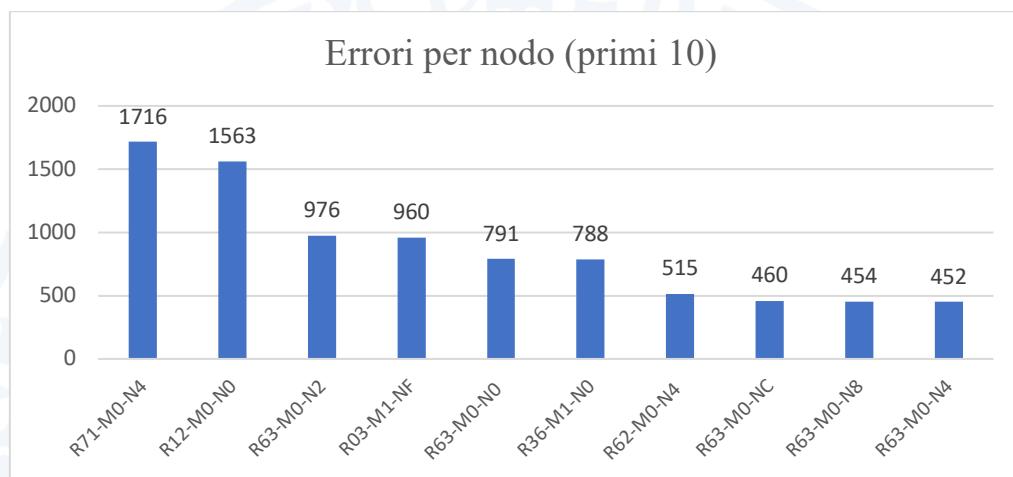
I dati relativi all'analisi sono stati raccolti in un file di log che contiene 125624 righe, ognuna delle quali riguarda un errore fatale del sistema. Questi log sono stati precedentemente filtrati e formattati seguendo uno schema specifico, che comprende le seguenti informazioni per ciascun record:

- ❖ **Timestamp**: Indica il momento in cui si è verificato l'errore fatale.
- ❖ **Nodo di origine del failure**: Identifica il nodo del sistema da cui è partito l'errore fatale.
- ❖ **Card di origine del failure**: Specifica la scheda del sistema associata all'errore fatale.
- ❖ **Messaggio di failure**: Descrive il tipo di errore fatale che si è verificato.

Di seguito è mostrata la prima entry del file di log:

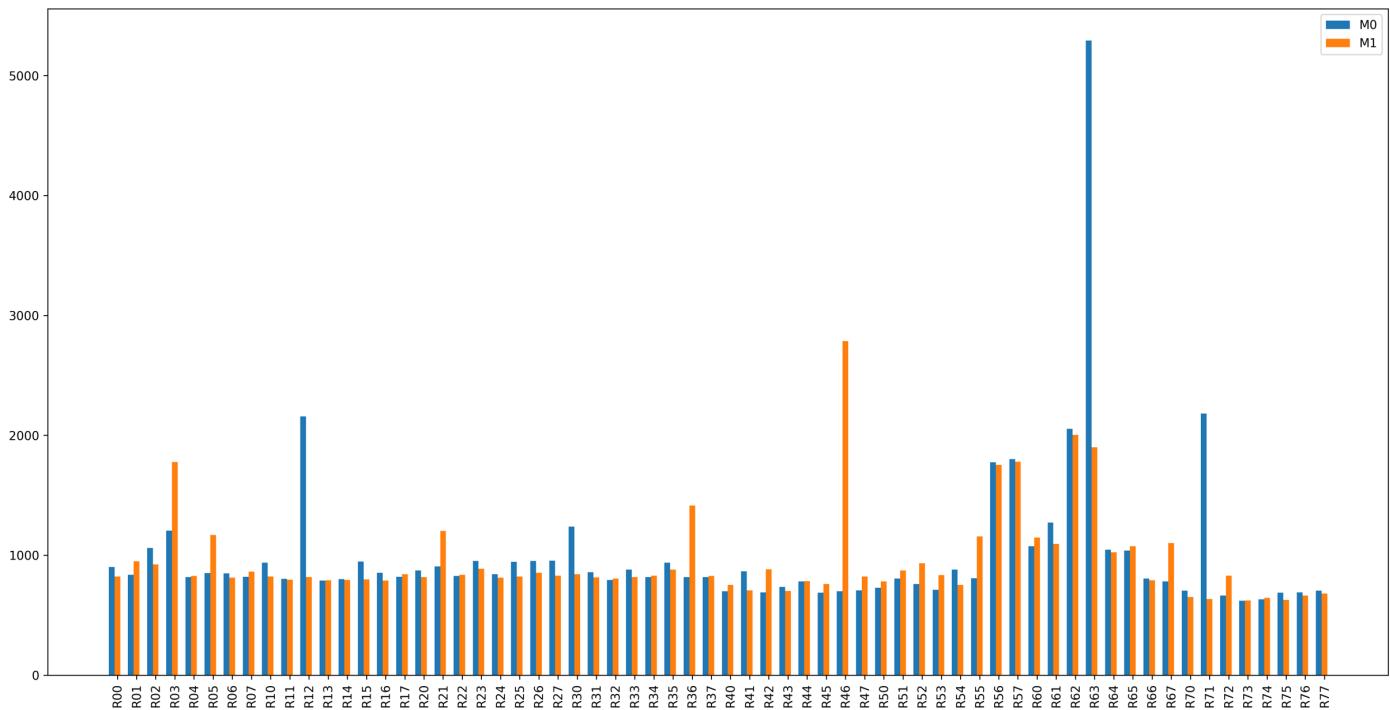
```
1. 1128621350 R00-M0-N0 J18-U01 Lustre mount FAILED : bglio2 : block_id : location
```

Di seguito sono mostrati il numero di errori per nodo e per rack.



Utilizzando lo script denominato contaRack.py, è stato condotto un processo di conteggio del numero di errori riscontrati per ciascun rack presente nel log, il quale costituisce la base della figura precedente.

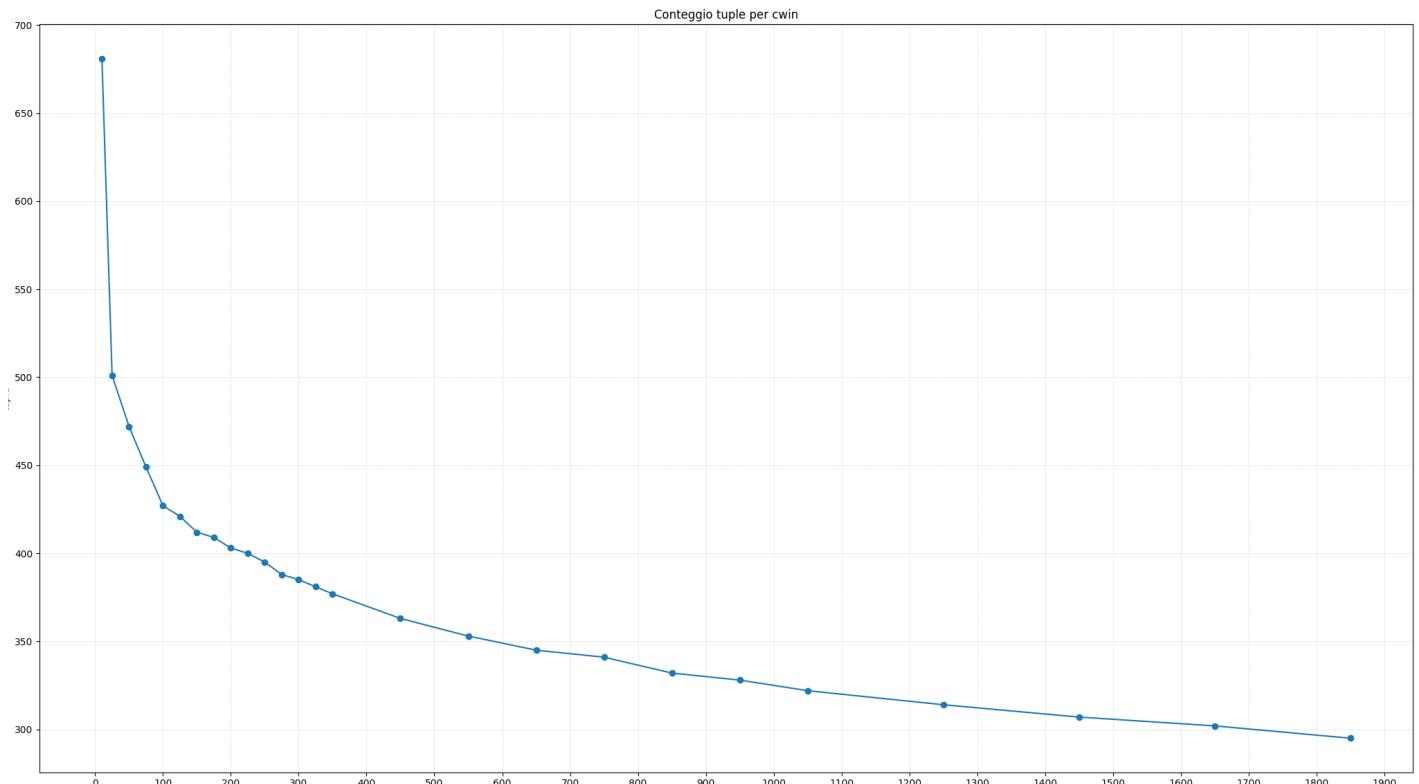
Successivamente, mediante l'impiego degli script Python scriptMid.py e scriptMid2.py, si è proceduto con il filtraggio del file di log relativo al Blue Gene (BGE), generando così dei log distinti per ciascun rack. In seguito, si è eseguito il conteggio del numero di errori sulle due midplane per ciascun log riferito ai rack e si è proceduto automaticamente alla creazione di un grafico, il quale è presentato nell'immagine successiva:



Gli errori sembrano abbastanza bilanciati tra le due midplane ad esclusione del nodo M63 che ha più errori su M0 ,M46 più nodi M1.

8.3.2 Data Manipulation

Si procede con il determinare la lunghezza della finestra di coalescenza. Si sceglie come valore ottimo della finestra di coalescenza, il valore di W immediatamente successivo al valore corrispondente al ginocchio della curva.

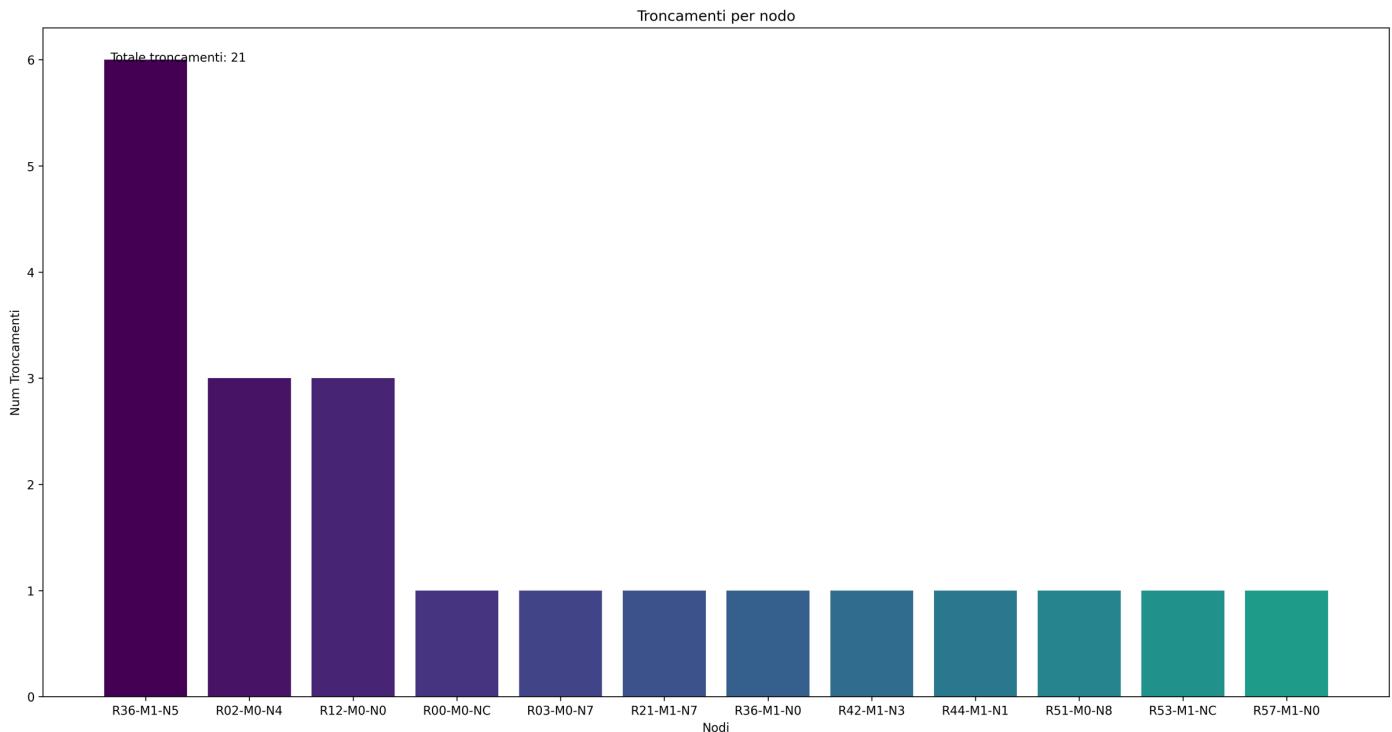


È stato quindi deciso di utilizzare una finestra di 200s poiché dopo di esso la pendenza diminuisce drasticamente.

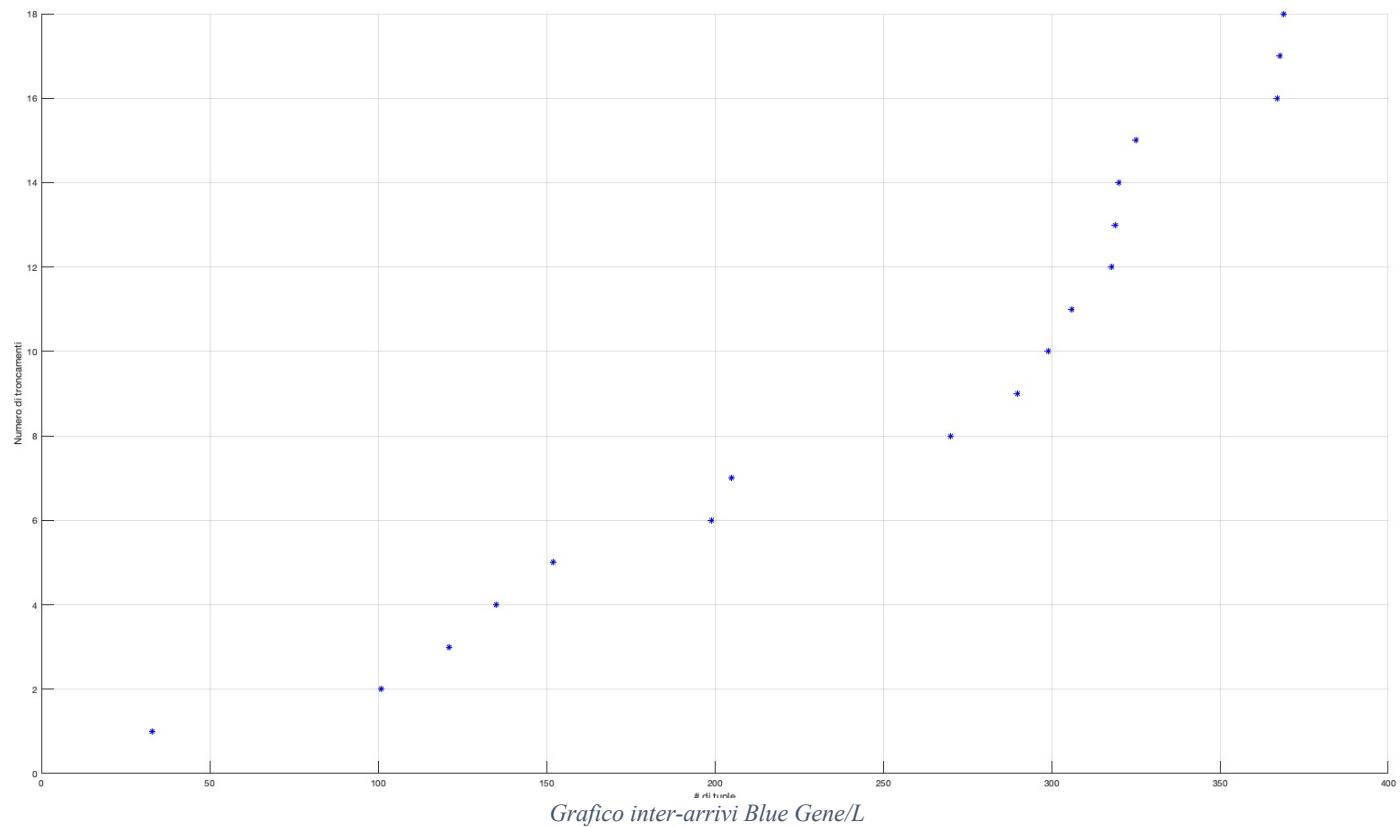
Tramite lo script *UStupling with CWIN.py* ricaviamo come prima le informazioni di lenght, starting points, interrival.

Verifichiamo la scelta della dimensione della finestra di coalescenza andando ad analizzare i troncamenti e le collisioni.

Utilizzando lo script python, anche usato per l'analisi di Mercury risultano 21 troncamenti.



Il numero di troncamenti sembrerebbe alquanto basso, e da tale grafico si può inoltre dedurre che il nodo R36-M1-N5 è quello che maggiormente è stato riscontrato a tra il termine di una tupla e l'inizio di altre. Andando a graficare gli inter-arrivi che sono maggiori del 50% della dimensione della finestra di coalescenza, sembrerebbe che la maggior parte dei troncamenti potrebbero essersi verificati tra i 275 ed i 325 s. Nella figura successiva è mostrato tale grafico.



Per quanto riguarda le collisioni lo script eseguito precedente ha calcolato 267 collisioni, si è quindi deciso di verificare le collisioni di diverse tipologie di card.

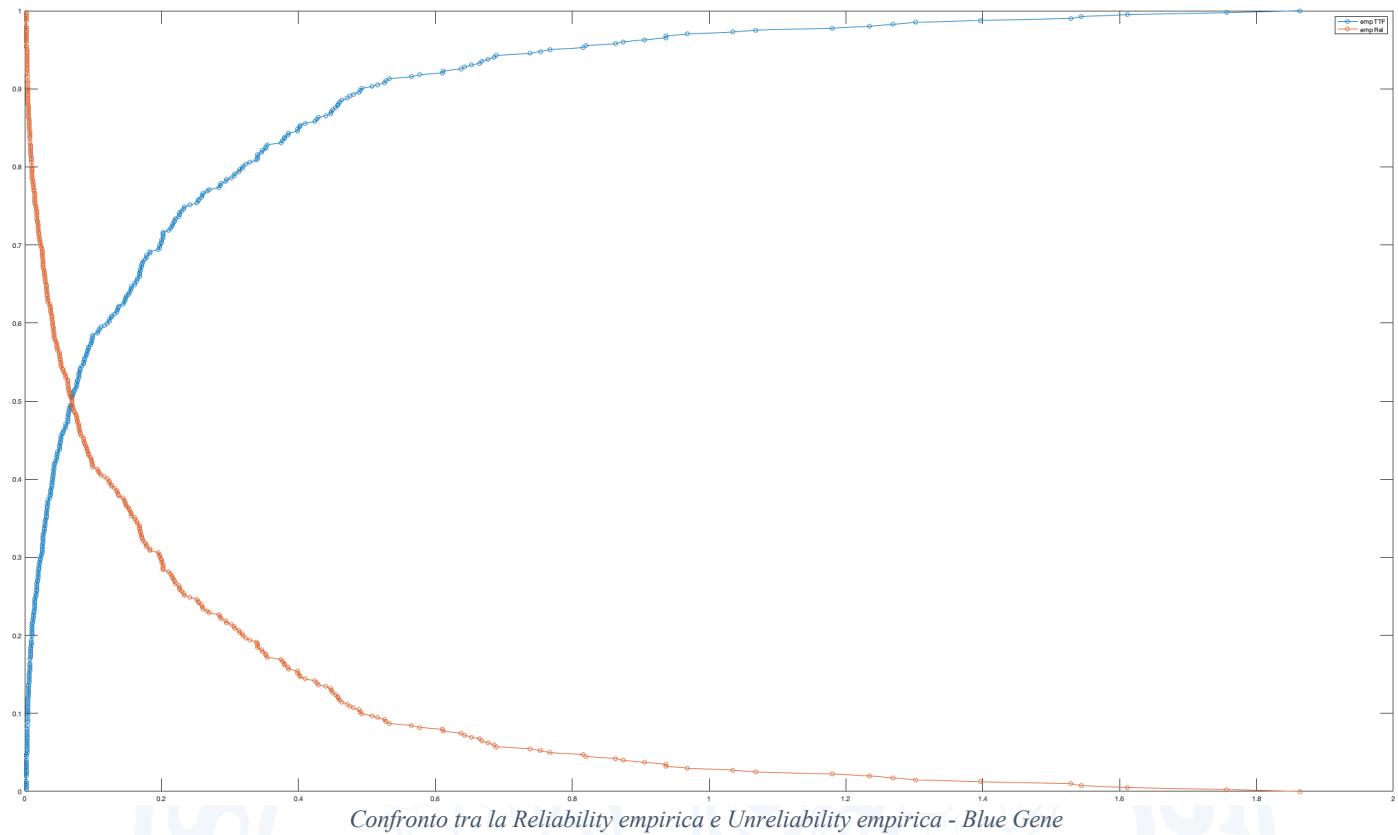
Da tale analisi risultano 39 collisioni di diversa tipologia di card. Nel file *collisioni_BGL_cardtype_per_tipologia.txt* sono presenti le collisioni riscontrate.

```
19: ['computation' 'I-O']
47: ['computation' 'I-O']
55: ['I-O' 'computation']
200: ['I-O' 'computation']
250: ['computation' 'I-O']
251: ['computation' 'I-O']
257: ['computation' 'I-O']
258: ['computation' 'I-O']
259: ['computation' 'I-O']
260: ['computation' 'I-O']
261: ['computation' 'I-O']
262: ['computation' 'I-O']
263: ['computation' 'I-O']
265: ['computation' 'I-O']
275: ['computation' 'I-O']
289: ['computation' 'I-O']
298: ['computation' 'I-O']
307: ['computation' 'I-O']
315: ['computation' 'I-O']
316: ['computation' 'I-O']
337: ['computation' 'I-O']
```

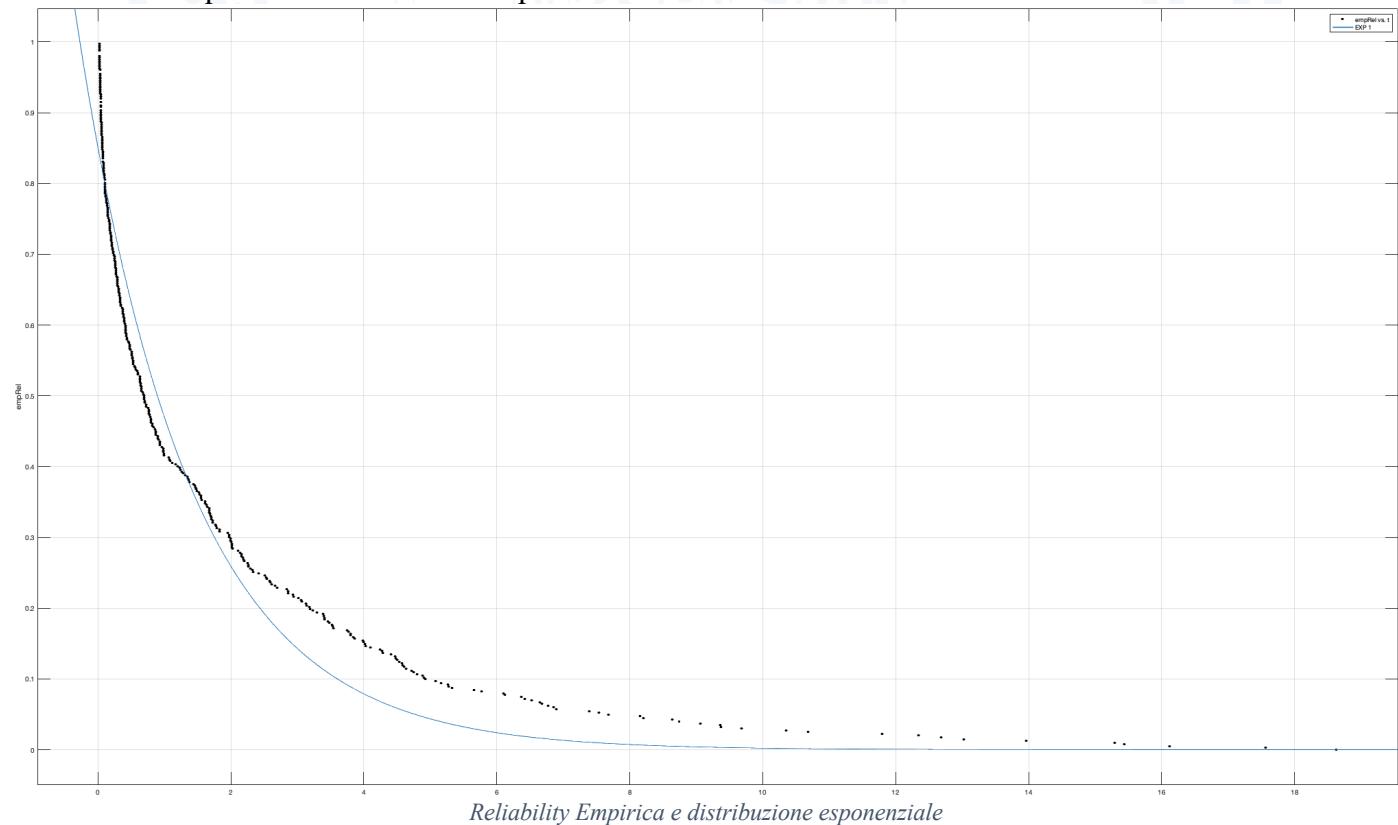
Tali risultati mostrano che la dimensione di 200s sia un valore ragionevole.

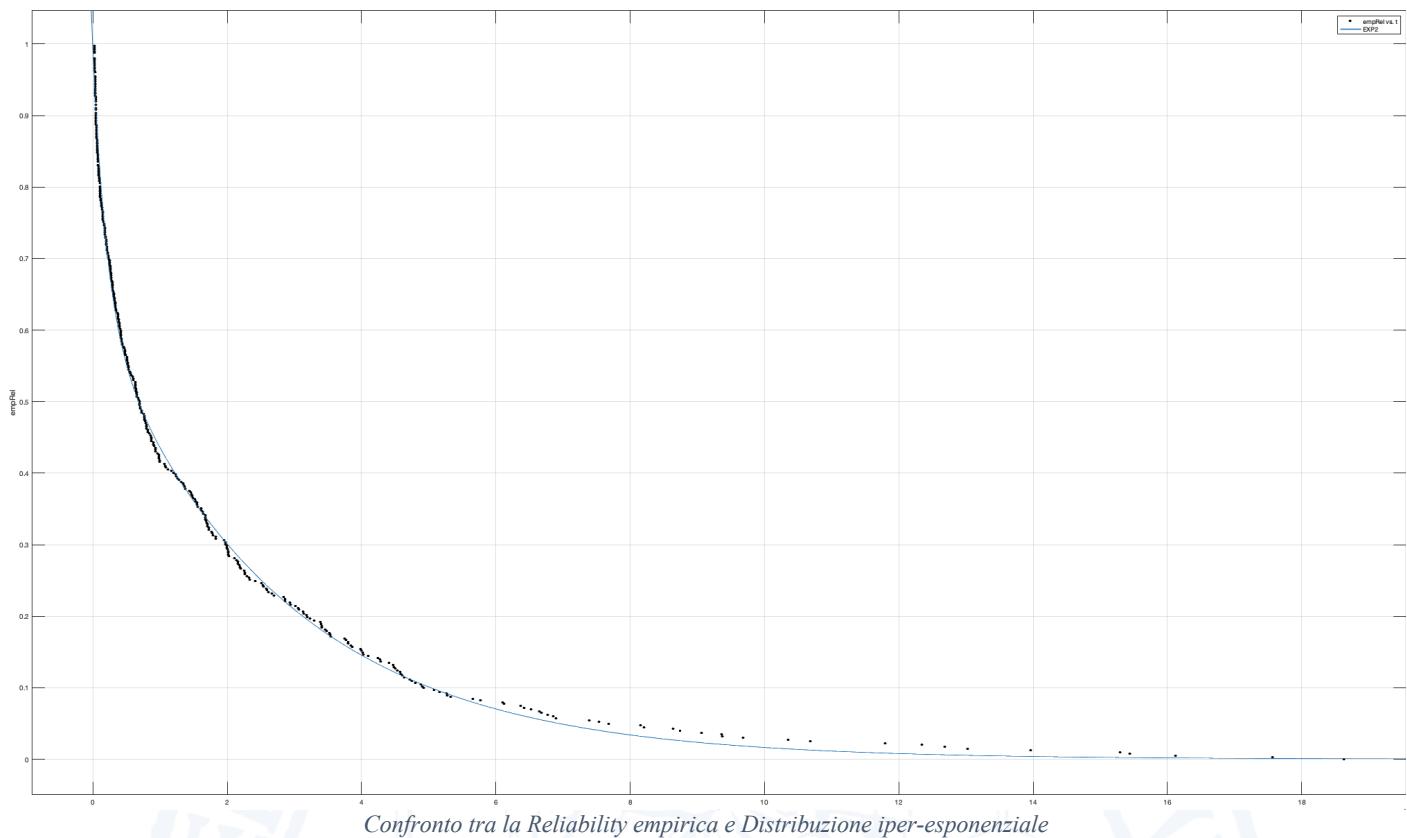
8.3.3 Data Analysis

Si procede in maniera analoga a quanto fatto con l'analisi di Mercury.

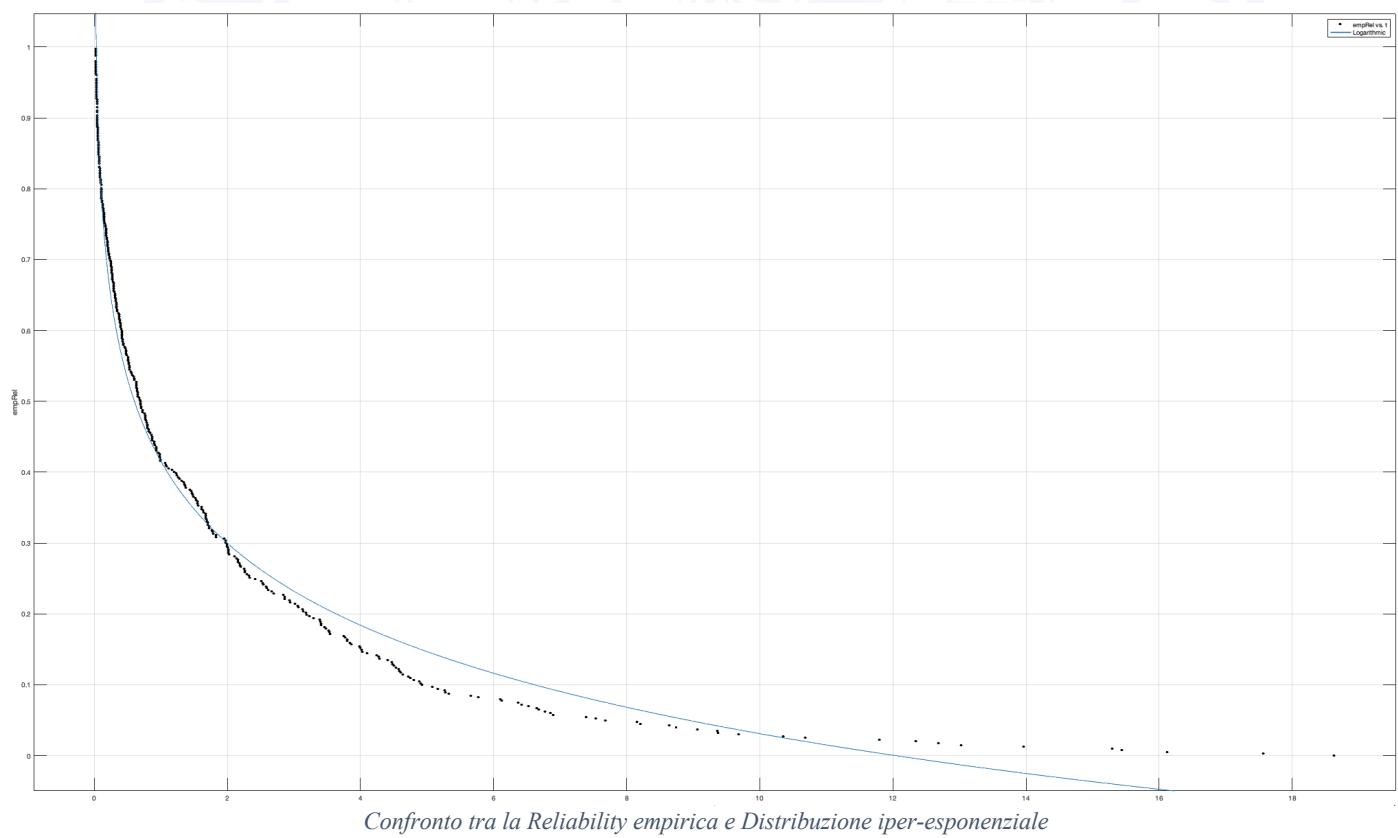


Si effettua dunque il confronto della empRel con le distribuzioni notevoli.





Confronto tra la Reliability empirica e Distribuzione iper-esponenziale



Confronto tra la Reliability empirica e Distribuzione iper-esponenziale

Di seguito sono presentati i parametri di qualità del fitting:

Fit name	Data	Fit type	R-square	SSE	DFE	Adj R-sq	RMSE	# Coeff
EXP 1	empRe...	exp1	0.95633	1.4006	392	0.95622	0.059774	2
EXP2	empRe...	exp2	0.99784	0.069291	390	0.99782	0.013329	4
Logarithmic	empRe...	log	0.98936	0.34131	392	0.98933	0.029507	2

Dalle analisi visive sembrerebbe migliore la distribuzione iper-esponenziale.

Come ulteriore verifica si effettua un *Kolmogorov-Smirnov test* utilizzando il seguente comando su matlab:

```
1. [h,p,k] = kstest2(empRel,fittedmodel(t))
```

Di seguito sono riportati i risultati conseguiti :

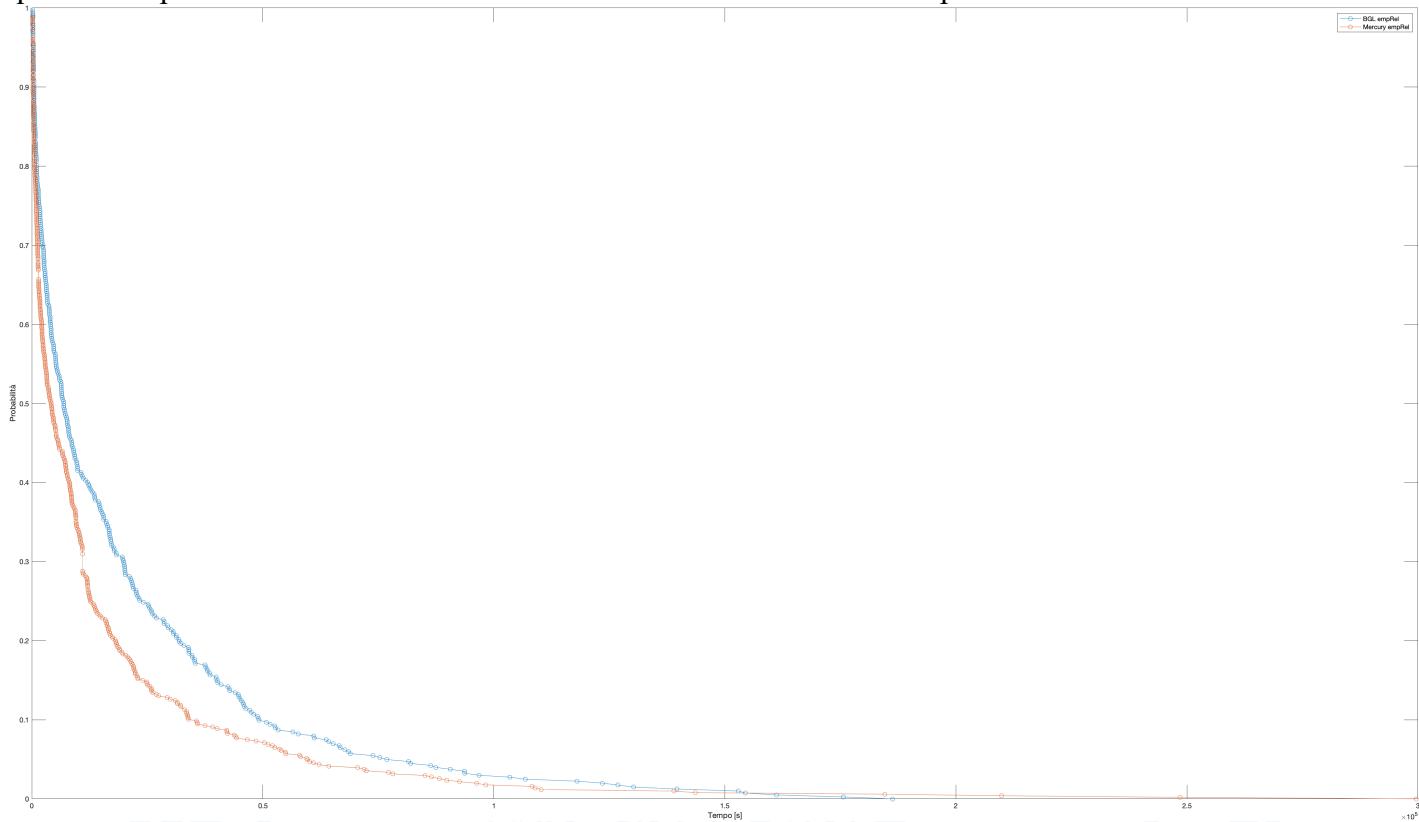
	h	p	k
Esponenziale	1	3.3507e-04	0.1472
Iper-esponenziale	0	0.8954	0.0406
Logaritmica	0	0.6189	0.0533

In conclusione, considerando la somiglianza tra la curva empirica e la distribuzione iper-esponenziale, è ragionevole supporre che la reliability del sistema segua un andamento che può essere descritto efficacemente mediante tale distribuzione.

Dalle analisi condotte su entrambi i sistemi si può intuire che probabilmente la reliability dei sistemi dipende principalmente da **fallimenti riconducibili all'hardware** (vista la grande compatibilità con la distribuzione iper-esponenziale).

8.4 Confronto Reliability tra Mercury e Blue Gene

Le funzioni di reliability fornite per i sistemi Mercury e Blue Gene consentono di confrontare l'affidabilità dei due sistemi. Confrontando le curve di reliability generate per Mercury e Blue Gene, è possibile determinare quale sistema dimostri una maggiore affidabilità in termini probabilistici, fornendo così un quadro comparativo dell'affidabilità dei due sistemi nell'intervallo di tempo considerato.



Dalla figura il sistema Blue Gene risulta più affidabile rispetto al sistema Mercury.

Per confermare tale assunzione viene calcolato approssimativamente l'integrale delle due reliability tramite Matlab.

```

clear all;
close all;
clc;

BGL_interarrivals= importdata("BGL/interarrivals.txt");
Mercury_interarrivals= importdata("Mercury/interarrivals.txt");

% Calcola le empRel per il set di dati BGL
[y_bgl, t_bgl] = cdfcalc(BGL_interarrivals);
empTTF_bgl = y_bgl(2:end);
empRel_bgl = 1 - empTTF_bgl;

% Calcola le empRel per il set di dati Mercury
[y_mercury, t_mercury] = cdfcalc(Mercury_interarrivals);
empTTF_mercury = y_mercury(2:end);
empRel_mercury = 1 - empTTF_mercury;

% Confronto delle empRel
plot(t_bgl, empRel_bgl, '-o', t_mercury, empRel_mercury, '-o');
xlabel('Tempo [s]');
ylabel('Probabilità');
legend('BGL empRel', 'Mercury empRel');

% Calcola l'integrale delle empRel per il set di dati BGL
integral_bgl = trapz(t_bgl, empRel_bgl);

```

```
% Calcola l'integrale delle empRel per il set di dati Mercury
integral_mercury = trapz(t_mercury, empRel_mercury);

disp(['Integrale dell affidabilita per BGL: ', num2str(integral_bgl)]);
disp(['Integrale dell affidabilita per Mercury: ', num2str(integral_mercury)]);
```

$$MTTF_M = \int_0^{+\infty} R_{empMercury} = 13559.6105$$

$$MTTF_{BG} = \int_0^{+\infty} R_{empBlueGene/L} = 18817.3495$$

Tale calcolo conferma che il supercalcolatore Blue Gene risulta maggiormente affidabile rispetto il supercalcolatore Mercury.

8.5 Analisi CWIN tra nodi (Mercury, BG/L) e categorie di errore (Mercury)

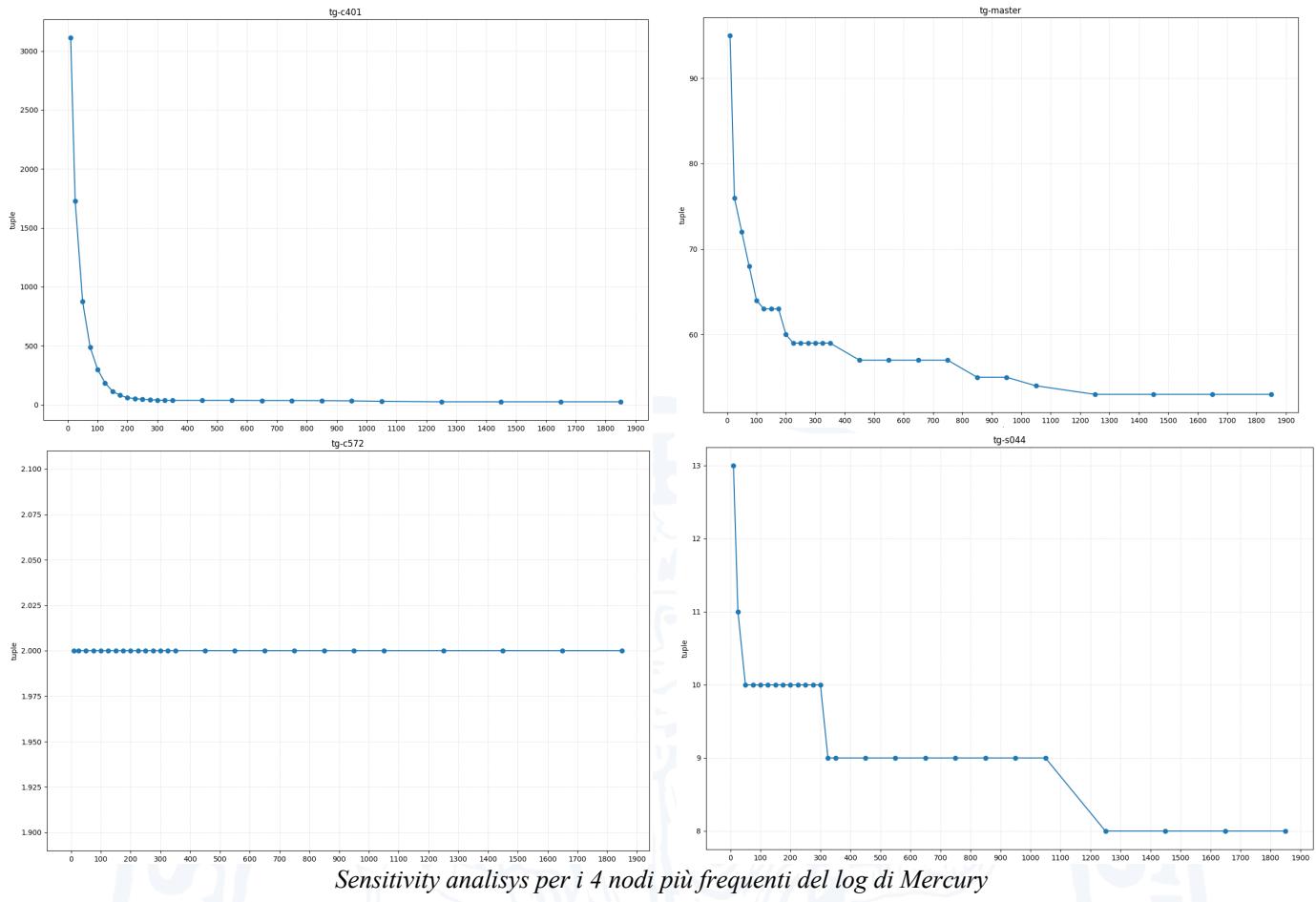
Nella presente sezione, l'obiettivo è determinare se le scelte effettuate riguardanti la durata della finestra di coalescenza per i due log analizzati possono essere estese a sottosistemi dei due supercalcolatori. Per raggiungere questo obiettivo, abbiamo adottato l'approccio seguente: abbiamo filtrato i due log forniti per nodo interessato e, nel caso del log di Mercury, anche per tipo di fallimento verificato. Dopo aver eseguito il filtraggio, abbiamo ripetuto l'analisi di sensibilità sui log associati ai sottosistemi al fine di identificare dimensioni appropriate per le finestre di coalescenza.

Per quanto riguarda Mercury, abbiamo scelto di condurre l'analisi di sensibilità sui seguenti nodi: tg-c401, tg-master, tg-c572, tg-s044, che rappresentano i 4 nodi con il maggior numero di voci all'interno del log. Tramite lo script USfilter.py si è proceduto quindi a filtrare i log per i vari nodi selezionati utilizzando il seguente comando:

- ```
1. python3 USfilter.py MercuryErrorLog.txt tg-master
2. python3 USfilter.py BGLErrorLog.txt R71-M0-N4
```

Generati i log filtrati per nodo, viene utilizzato lo script tupleCount\_func\_CWINpy.sh per ogni file di log per calcolare il numero di tuple al variare della dimensione della finestra temporale. Successivamente tali risultati vengono graficati automaticamente tramite lo script FilteredCwinGrafici.py .

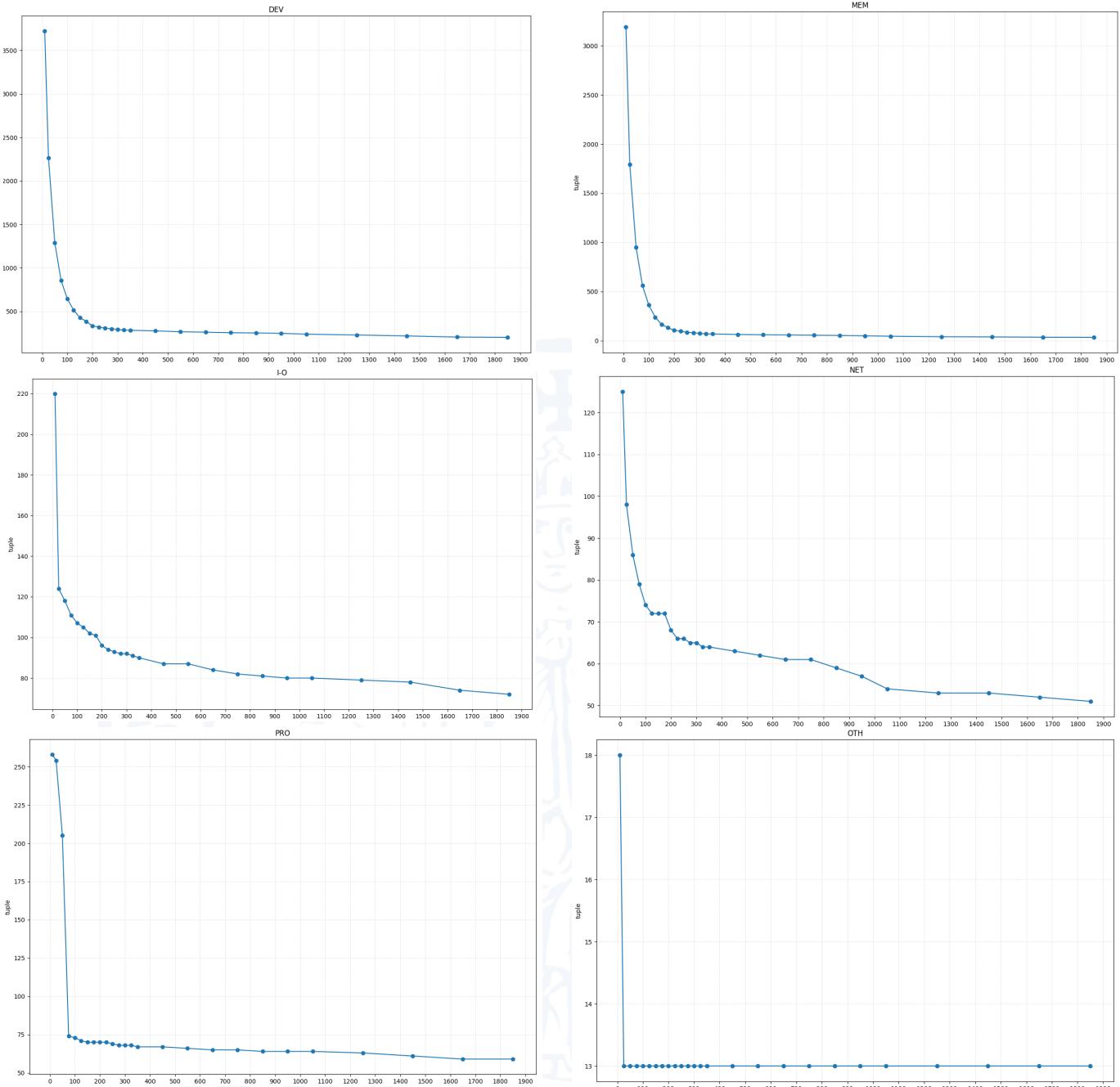
Di seguito sono mostrati i grafici ottenuti del sistema Mercury:



Sensitivity analysis per i 4 nodi più frequenti del log di Mercury

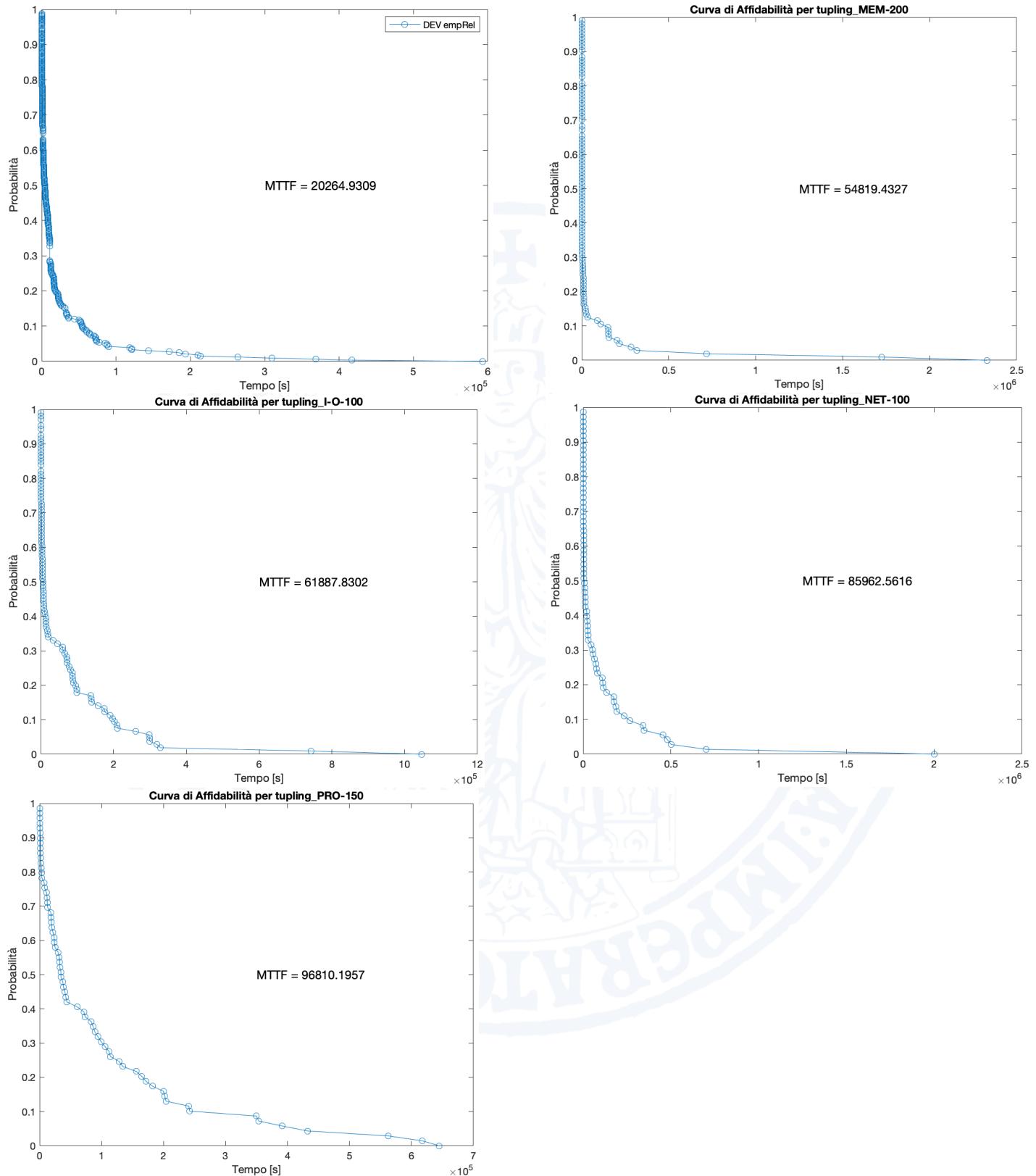
Dalla presente analisi emerge che per i primi due nodi, i quali manifestano un numero estremamente elevato di voci all'interno del log, si osserva una finestra di coalescenza approssimativamente pari a 200 secondi, avvicinandosi quindi a quella selezionata per l'analisi del log. Tuttavia, per gli altri due nodi, si osserva un risultato differente: nel caso del nodo tg-s044 è possibile selezionare una finestra di coalescenza molto più breve, mentre per il nodo tg-c572, che presenta esclusivamente voci isolate all'interno del log, la durata della finestra selezionata risulta insignificante. In generale, si può concludere che per i nodi che generano la maggior parte degli eventi registrati nel log, può essere utilizzata una finestra di coalescenza pari a quella adottata per l'intero sistema. Tuttavia, tale conclusione non si applica necessariamente ai sottosistemi che generano solo una piccola parte delle voci presenti.

Ripetiamo tale analisi per le diverse categorie:

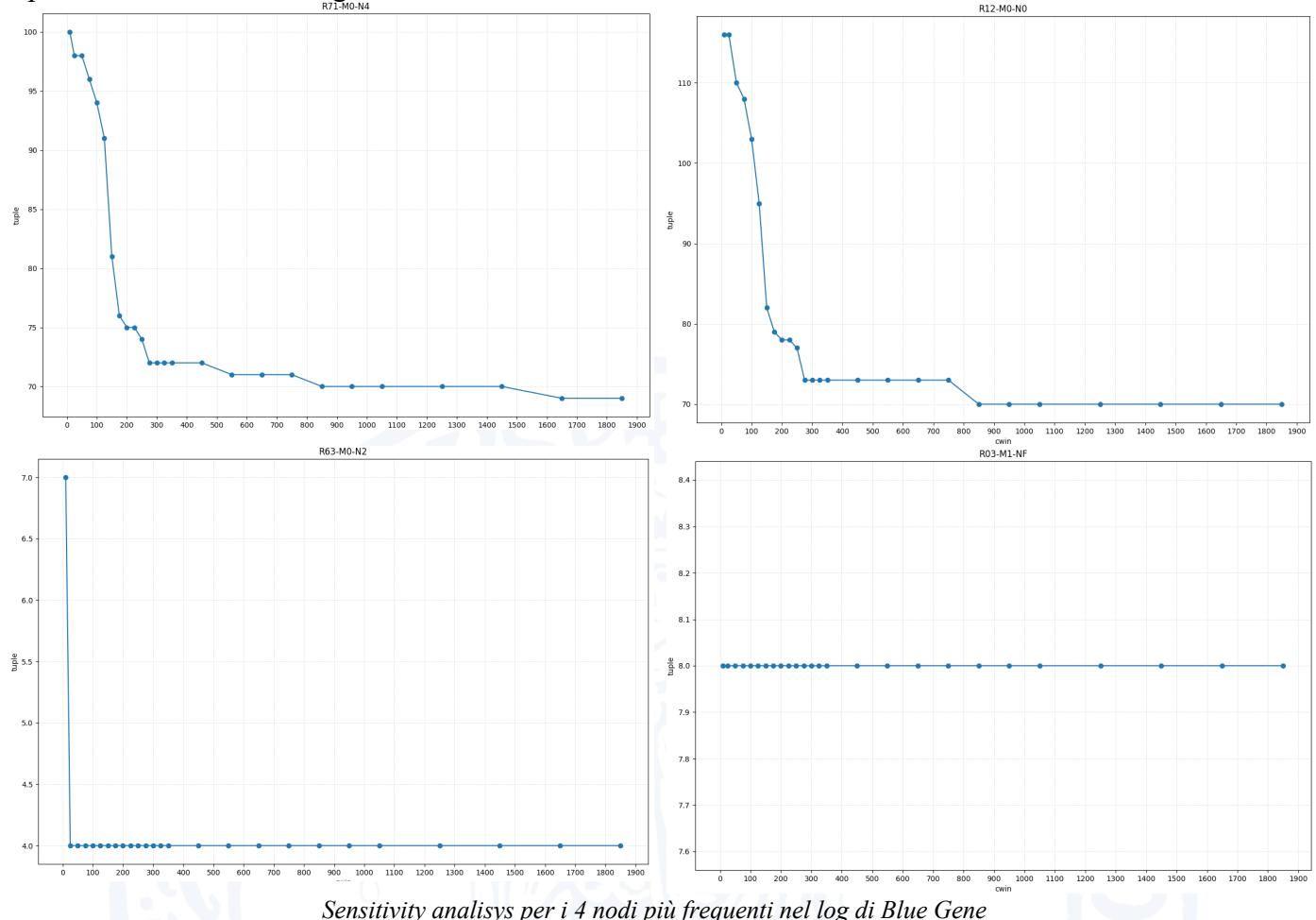


Il risultato ottenuto è coerente con quanto precedentemente affermato: per le categorie di errori più frequenti, come DEV e MEM, è possibile adottare la medesima durata della finestra di coalescenza del sistema complessivo, approssimativamente 200 secondi. Tale considerazione, tuttavia, non si estende al caso delle categorie di errori meno frequenti, come OTH o PRO, per le quali è opportuna la selezione di una finestra di coalescenza notevolmente più breve.

Si confrontano le reliability di ogni categoria andando a considerare le diverse dimensioni delle finestre di coalescenza (DEV e MEM 200s, PRO 150s, I-O e NET 100s. )



Si procede in maniera analoga all'analisi del sistema Blue Gene/L al variare dei nodi:

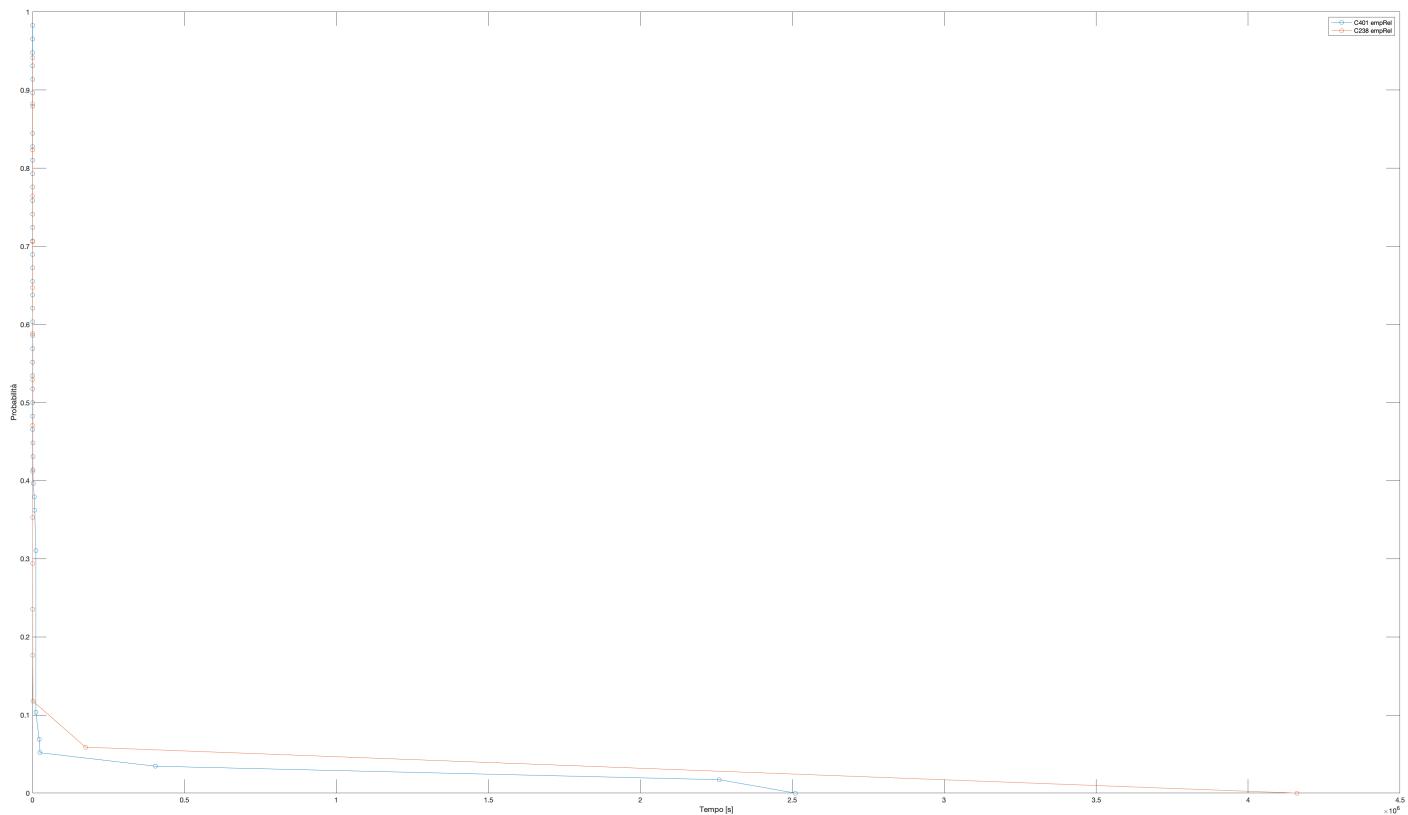


Sensitivity analysis per i 4 nodi più frequenti nel log di Blue Gene

I risultati ottenuti indicano che la scelta della durata della finestra di coalescenza per l'intero sistema non è applicabile in modo adeguato ai singoli nodi. Nel caso specifico di R71-M0-N4 e R12-M0-N0, risulta opportuno selezionare una durata più lunga, approssimativamente 290 secondi. Per quanto riguarda R63-M0-N2, è possibile optare per una finestra molto più breve di circa 50 secondi. Tuttavia, nel caso di R03-M1-NF, la durata della finestra non ha influenza poiché questo nodo genera esclusivamente voci isolate.

## 8.6 Confronto Reliability Mercury Nodi Computazionali

Si valuta in questo paragrafo la Reliability tra due nodi computazionali del sistema Mercury. In particolare tale confronto sarà effettuato tra i nodi computazionali più presenti ovvero TG-C401 e TG-C238, utilizzando una finestra di coalescenza di 200s per entrambi i nodi.



Da tale grafico le due reliability risultano molto simili.

## 8.7 Correlazione tra nodi e categorie di errore nel calcolatore Mercury

Si analizza il numero di errori dei 10 nodi più presenti nei log e li si suddividono in base alla categoria d'errore.

| Nodo             | DEV   | MEM   | IO   | NET  | PRO | OTH |
|------------------|-------|-------|------|------|-----|-----|
| <b>tg-c117</b>   | 263   | 5     | 0    | 0    | 0   | 0   |
| <b>tg-c238</b>   | 1071  | 0     | 230  | 3    | 0   | 0   |
| <b>tg-c242</b>   | 918   | 149   | 0    | 0    | 0   | 0   |
| <b>tg-c401</b>   | 50782 | 11558 | 0    | 0    | 0   | 0   |
| <b>tg-c572</b>   | 3176  | 845   | 0    | 0    | 0   | 0   |
| <b>tg-c648</b>   | 643   | 0     | 0    | 0    | 616 | 0   |
| <b>tg-c669</b>   | 257   | 10    | 0    | 0    | 0   | 0   |
| <b>tg-login3</b> | 0     | 0     | 381  | 1    | 0   | 0   |
| <b>tg-master</b> | 0     | 0     | 452  | 3639 | 0   | 0   |
| <b>tg-s044</b>   | 0     | 0     | 3208 | 2    | 0   | 14  |

Si trasforma tale matrice in percentuale in base al totale di entry del nodo:

| Nodo             | DEV         | MEM         | IO          | NET         | PRO         | OTH         |
|------------------|-------------|-------------|-------------|-------------|-------------|-------------|
| <b>tg-c117</b>   | 98,13432836 | 1,865671642 | 0           | 0           | 0           | 0           |
| <b>tg-c238</b>   | 82,13190184 | 0           | 17,63803681 | 0,23006135  | 0           | 0           |
| <b>tg-c242</b>   | 86,03561387 | 13,96438613 | 0           | 0           | 0           | 0           |
| <b>tg-c401</b>   | 81,45973693 | 18,54026307 | 0           | 0           | 0           | 0           |
| <b>tg-c572</b>   | 78,98532703 | 21,01467297 | 0           | 0           | 0           | 0           |
| <b>tg-c648</b>   | 51,07227959 | 0           | 0           | 0           | 48,92772041 | 0           |
| <b>tg-c669</b>   | 96,25468165 | 3,745318352 | 0           | 0           | 0           | 0           |
| <b>tg-login3</b> | 0           | 0           | 99,7382199  | 0,261780105 | 0           | 0           |
| <b>tg-master</b> | 0           | 0           | 11,04864336 | 88,95135664 | 0           | 0           |
| <b>tg-s044</b>   | 0           | 0           | 99,50372208 | 0,062034739 | 0           | 0,434243176 |

Calcolando la media per i nodi della stessa tipologia:

| Nodo                  | DEV   | MEM  | IO    | NET   | PRO  | OTH  |
|-----------------------|-------|------|-------|-------|------|------|
| <b>Computazionali</b> | 82,01 | 8,45 | 2,52  | 0,03  | 6,99 | 0,00 |
| <b>Login</b>          | 0,00  | 0,00 | 99,74 | 0,26  | 0,00 | 0,00 |
| <b>Master</b>         | 0,00  | 0,00 | 11,05 | 88,95 | 0,00 | 0,00 |
| <b>Storage</b>        | 0,00  | 0,00 | 99,50 | 0,06  | 0,00 | 0,43 |

Si può verificare che i nodi di calcolo sono comunemente esposti a errori di tipo DEV o MEM, mentre il nodo master risulta prevalentemente coinvolto in errori di tipo NET. D'altra parte, sia i nodi di tipo Login che quelli di tipo Storage mostrano una predominante incidenza (oltre il 99%) di errori di tipo IO.