



UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
Scuola Politecnica e delle Scienze di Base
Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione
(DIETI)

Corso di Laurea Magistrale in Ingegneria Informatica

Elaborato di Impianti di Elaborazione

Studenti:

Fabio Fiumara [M63001795]
Marcello Esposito [M63001768]
Lorenzo Aniello Alessandrella [M63001782]

Anno Accademico 2025/2026

Indice

1. Benchmark	1
1.1 Introduzione	2
1.2 Configurazione di Nbody.....	2
1.3 Dataset	3
1.3.1 Risultati 10K corpi	3
1.3.2 Risultati 100K corpi	4
1.3.3 Risultati 1M corpi	5
1.3.4 Conclusioni	6
1.4 Intervalli di confidenza.....	6
1.4.1 10K corpi	6
1.4.2 100K corpi	7
1.4.3 1M corpi	7
1.4.4 Conclusioni	8
1.5 Test d'ipotesi: Paired T-Test	8
1.5.1 10K corpi	9
1.5.2 100K corpi	10
1.5.3 1M corpi	11
1.6 Conclusioni	12
2. PCA & Clustering.....	13
2.1 Introduzione	13
2.2 Prefiltraggio	14
2.3 PCA.....	14
2.4 Clustering.....	15
2.5 Clustering senza PCA	16
3. Web Server	17
3.1 Capacity Test	18
3.1.1 Introduzione	18
3.1.2 DUT.....	18
3.1.3 Test plan.....	18
3.1.4 High Level analysis	19
3.1.5 Low Level analysis.....	20
3.1.5 Fairness Index.....	22
3.1 Workload Characterization	23

3.1.1 Introduzione	24
3.1.2 Setup	24
3.1.3 Fase 1: caratterizzazione workload reale	25
HL'	25
LL'	26
3.1.4 Fase 2: LL'c	28
3.1.5 Fase 3: Data validation	29
4. Regression	30
Introduzione.....	31
1. EXP	31
2. os	35
3. Heap	44
5. Design of Experiments.....	47
Introduzione.....	48
Configurazione	48
Analisi di Importanza	49
Analisi di Significatività	51
Analisi Normalità.....	51
Analisi Omoschedasticità	53
Conclusioni	54
6. Esercizi Dependability	54
Esercizio 1	55
Esercizio 2	56
Esercizio 3	59
Esercizio 4	61
Esercizio 5	64
7. FFDA: analisi file di log	68
7.1 Mercury	68
7.1.1 Analisi globale	69
1.Esponenziale ().....	70
2.Iperesponenziale ()	71
3.Weibull ()	72
7.1.2 Categorie	76
7.1.3 Nodi.....	78
7.2 BG/L.....	82
7.2.1 Analisi globale	82

1.Esponenziale ().....	83
2.Iperesponenziale ()	84
3.Weibull ()	85
7.2.2Cards.....	88
7.2.3Nodi.....	91
7.3Conclusioni.....	94

1. Benchmark

1.1 Introduzione

Un benchmark è un particolare test workload tramite il quale siamo in grado di confrontare le prestazioni di sistemi differenti. Nell'esempio considerato è stato usato Nbody; esso è un particolare tipo di simulazione utile a studiare il movimento di un gruppo di corpi celesti che interagiscono attraverso forze. In questa situazione, il SUT risulta essere il processore, mentre il CUS è la Floating Point Unit. Nell'analisi effettuata sono state confrontate le prestazioni di due PC, le cui specifiche sono presentate nella tabella seguente:

	ASUS Vivobook Flip 14	Lenovo Ideapad 3 15alc6
Processore	Intel Core i3-10110U	Ryzen 7 5700U
Frequenza base	2.10 GHz	1.8 GHz
RAM	8 GB	18 GB
OS	Microsoft Windows 11 Home	Arch Linux

Il benchmark è stato poi eseguito, su entrambi i sistemi, andando ad utilizzare una macchina virtuale Kali Linux 2025 impostando 4GB di RAM e 4 core.

1.2 Configurazione di Nbody

Per ottenere i risultati di output, dopo aver compilato i file, utilizziamo il seguente comando da terminale, ponendoci ovviamente prima di tutto nella directory nella quale è posto il progetto:

```
./launch_nbody.sh -r $REPETITIONS -n $BODIES >> output_nbody_n.txt
```

In particolare, r specifica il numero di ripetizioni che andiamo ad effettuare per ogni singola simulazione. Abbiamo scelto di effettuare cinque ripetizioni, in quanto questo valore permette di ottenere un buon compromesso tra l'accuratezza dei risultati e il tempo impiegato per eseguire il benchmark. Abbiamo così la possibilità di andare a valutare le deviazioni standard e gli intervalli di confidenza, migliorando l'intera affidabilità delle analisi statistiche senza un numero eccessivo di misurazioni. Il valore della singola osservazione sarà poi dato dalla media delle cinque ripetizioni della simulazione. Il valore n specifica invece il numero di corpi simulati, utilizzato anche nella nomenclatura dei file di output in modo tale da dividere i risultati ottenuti in base al numero di corpi. Lo script è stato avviato prima con 10000, poi con 100000 e infine con 1000000 di corpi; in questo modo abbiamo la possibilità di simulare vari tipi di carico sul sistema che ci permettono di eseguire un confronto significativo riducendo l'importanza della variabilità casuale.

Per ogni singolo valore di n sono stati raccolti 40 campioni. È stato scelto di effettuare la raccolta di 40 campioni per due motivi fondamentali; il primo, quello più importante, è quello legato al Teorema del Limite Centrale, secondo il quale se la dimensione dei campioni è sufficientemente grande (40 campioni è un valore accettabile) la distribuzione delle medie campionarie è approssimabile con una normale standard in maniera totalmente indipendente dalla distribuzione iniziale dei dati, in modo tale da poter applicare metodi statistici basati proprio su questo particolare tipo di distribuzione. Il grande vantaggio

apportato da questo teorema è la possibilità di sostituire la deviazione standard della popolazione con quella campionaria senza influenzare l'accuratezza in maniera significativa, in quanto la deviazione standard campionaria sarà una stima affidabile della popolazione, solo grazie al fatto che abbiamo considerato un numero molto elevato di campioni. Tramite il teorema considerato abbiamo anche la possibilità di calcolare gli intervalli di confidenza affidabili per la media campionaria, migliorando l'affidabilità dei risultati, in quanto aumentando il numero di osservazioni si riduce la variabilità statistica e l'errore standard, in modo tale da ottenere stime più precise dei parametri del sistema.

1.3 Dataset

Il secondo motivo fondamentale per il quale abbiamo scelto 40 campioni è un'analisi della dimensione campionaria che ci permette di approssimare la media e la varianza campionarie con quelle dell'intera popolazione; il nostro studio prevede allora di verificare che la differenza in valore assoluto tra la media campionaria e quella della popolazione sia minore di una certa quantità E su un intervallo di confidenza del 95%. A questo punto valutiamo la dimensione campionaria come:

$$n = \left(\frac{z_{\alpha/2} s}{E} \right)^2$$

In cui s è la deviazione standard campionaria mentre $z_{\alpha/2}$ corrisponde al valore della funzione z ottenuto dalla z-table per un valore di α pari a 0.05. Verificando i risultati ottenuti considerando diversi valori dell'errore di stima, cioè 5%, 8% e 10%, per i due sistemi:

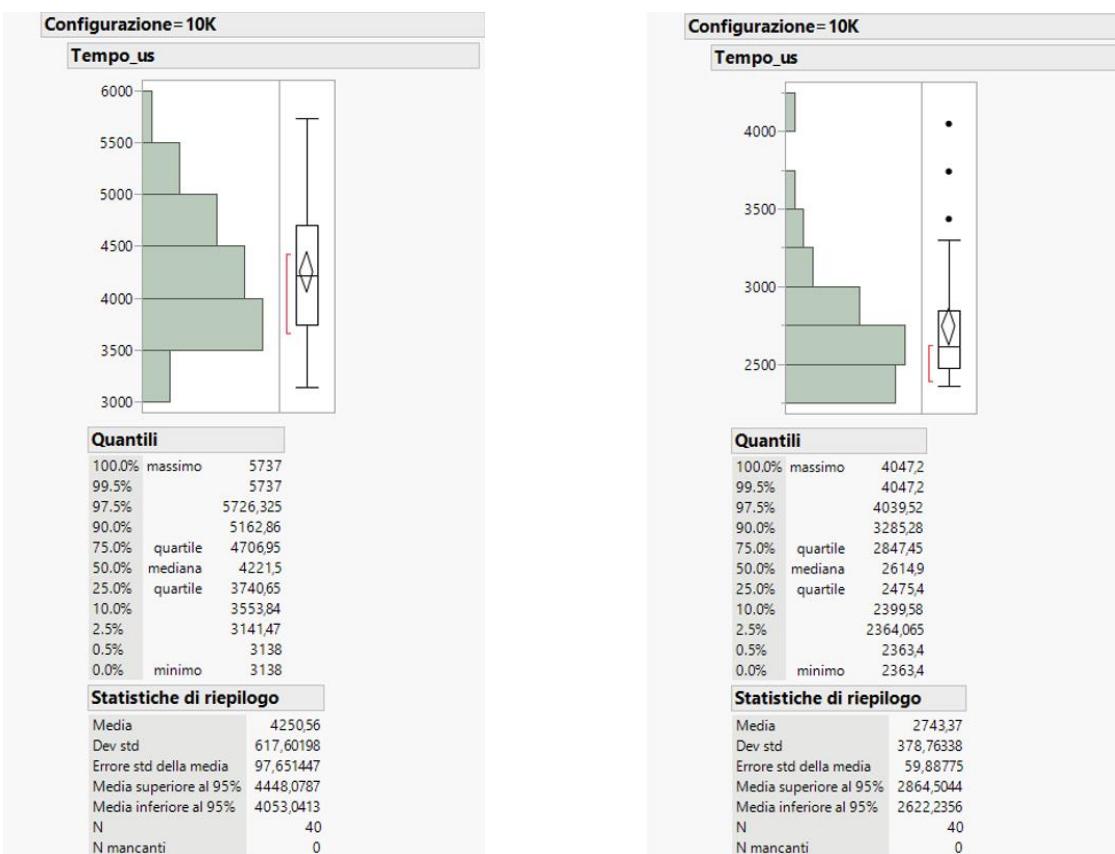
RISULTATI 10K CORPI	ASUS	Lenovo
5%	33	30
8%	13	12
10%	9	8

RISULTATI 100K CORPI	ASUS	Lenovo
5%	26	14
8%	11	6
10%	7	4

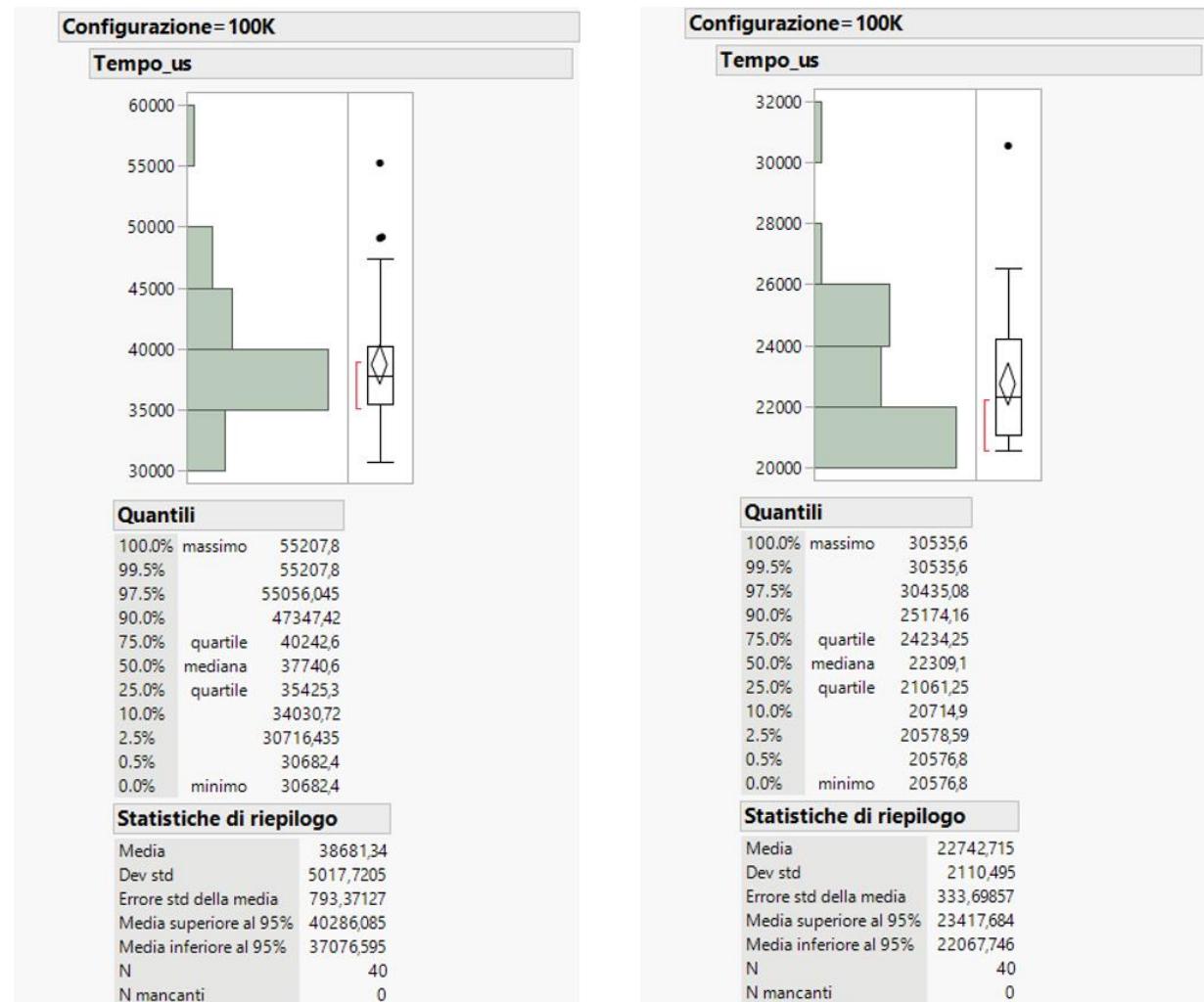
RISULTATI 1M CORPI	ASUS	Lenovo
5%	15	2
8%	6	1
10%	4	1

Possiamo facilmente verificare come i 40 campioni scelti precedentemente permettono di rispettare qualunque valore dell'errore di stima per tutte e tre le configurazioni di Nbody su entrambi i computer. Per cui, decidiamo di utilizzare un errore di stima del 5%.

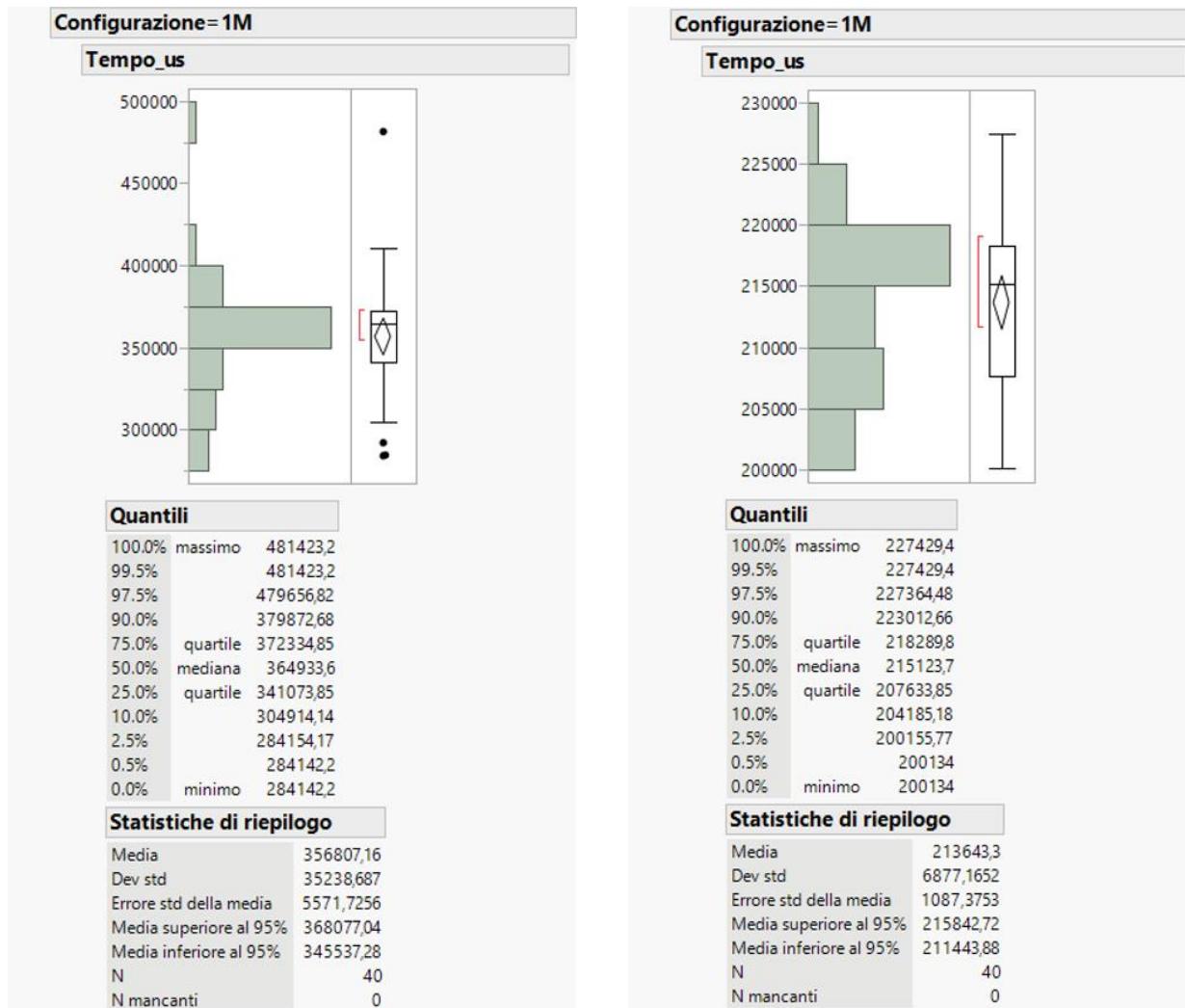
1.3.1 Risultati 10K corpi



1.3.2 Risultati 100K corpi



1.3.3 Risultati 1M corpi



1.3.4 Conclusioni

Sotto ognuno degli istogrammi presentati troviamo i valori estremi, i quartili e i percentili per ogni sistema. Possiamo allora facilmente notare come per tutti e tre i tipi di carico abbiamo una maggiore dispersione dei tempi di esecuzione, ottenibile guardando le deviazioni standard, nel sistema ASUS rispetto al Lenovo, differenza che si accentua in particolare nella situazione di carico alto, dove nel sistema ASUS abbiamo una fortissima dispersione. Anche per quanto riguarda i tempi medi di risposta il sistema Lenovo si comporta sempre sostanzialmente meglio del sistema ASUS.

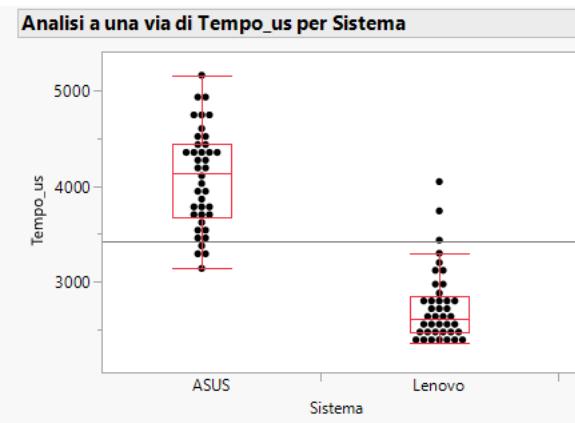
1.4 Intervalli di confidenza

In seguito, abbiamo effettuato un'analisi dei box plot e degli intervalli di confidenza per entrambi i sistemi, ottenendo i seguenti risultati:

1.4.1 10K corpi

Intervalli di confidenza				
Parametro	Stima	CI inferiore	CI superiore	1-Alfa
Media	4250,56	4053,041	4448,079	0,950
Dev std	617,602	505,9155	793,0229	0,950

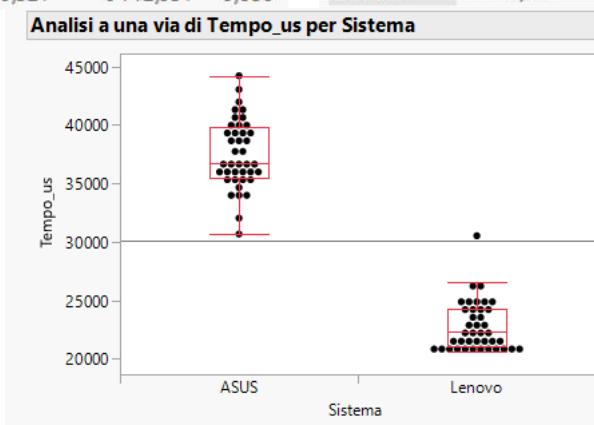
Intervalli di confidenza				
Parametro	Stima	CI inferiore	CI superiore	1-Alfa
Media	2743,37	2622,236	2864,504	0,950
Dev std	378,7634	310,2682	486,3456	0,950



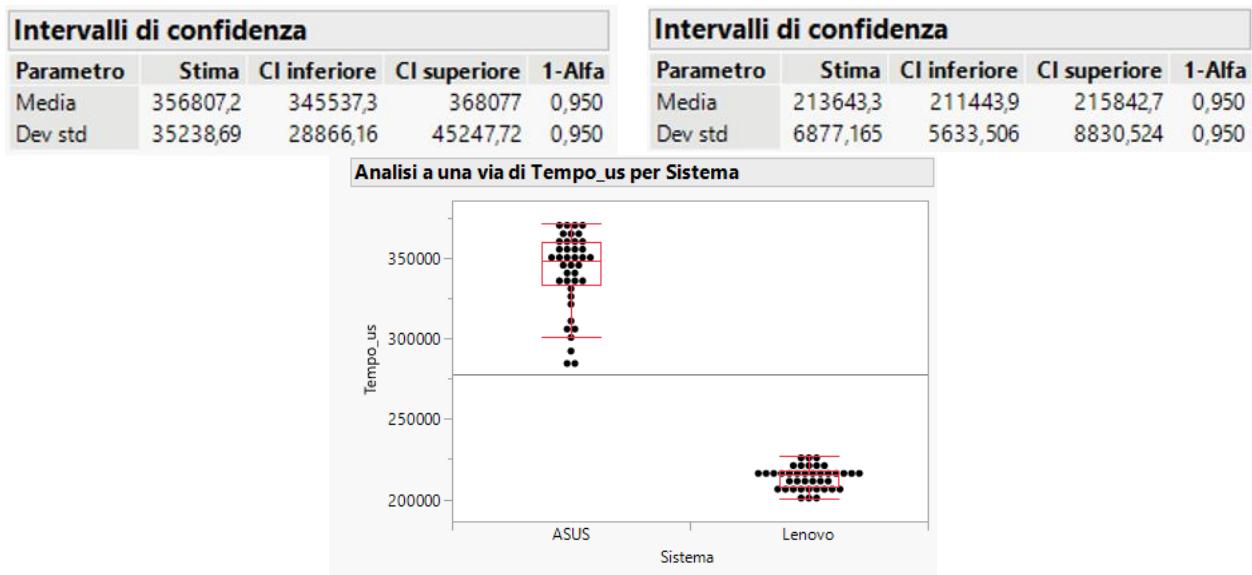
1.4.2 100K corpi

Intervalli di confidenza				
Parametro	Stima	CI inferiore	CI superiore	1-Alfa
Media	38681,34	37076,6	40286,08	0,950
Dev std	5017,72	4110,321	6442,931	0,950

Intervalli di confidenza				
Parametro	Stima	CI inferiore	CI superiore	1-Alfa
Media	22742,72	22067,75	23417,68	0,950
Dev std	2110,495	1728,835	2709,951	0,950



1.4.3 1M corpi



1.4.4 Conclusioni

Notiamo che l'intervallo di confidenza per il Lenovo è decisamente più stretto e inferiore rispetto a quello dell'ASUS, confermando quindi le sue performance migliori e una maggiore consistenza. L'intervallo di confidenza rappresenta un range nel quale ci aspettiamo che si trovi la media reale; quindi, un intervallo stretto indica una maggiore concentrazione dei dati intorno alla media, indicando una coerenza maggiore nei risultati. Infine, possiamo intuire facilmente la differenza tra le distribuzioni dei due sistemi già andando ad analizzare graficamente i due box plot per ogni tipo di carico.

1.5 Test d'ipotesi: Paired T-Test

Un test d'ipotesi è una certa procedura usata per valutare se ci sono sufficienti evidenze per poter supportare un'ipotesi o respingerla riguardo a una determinata popolazione. Tramite questa tecnica abbiamo la possibilità di determinare se una differenza nei dati osservati risulta essere significativa o meno, dunque casuale.

Nel nostro caso utilizziamo il test d'ipotesi per stabilire se esiste una differenza significativa nelle prestazioni dei sistemi considerati, cioè se effettivamente uno dei due risultati essere più veloce dell'altro o più stabile studiando il tempo di esecuzione, cercando quindi di confermare o smentire i test visivi effettuati precedentemente.

Per fare ciò definiamo due ipotesi:

H_0 (ipotesi nulla): afferma che non ci sia alcuna differenza significativa nei dati e che qualunque variazione eventualmente presente è casuale.

H_1 (ipotesi alternativa): rappresenta quello che vogliamo andare a verificare; quindi, nel nostro caso è l'ipotesi per la quale esiste una differenza significativa nelle prestazioni dei due sistemi.

Nella nostra situazione effettuiamo un Paired T-Test, il quale permette di lavorare con varianze campionarie stimate; è paired perché stiamo confrontando le prestazioni dello stesso

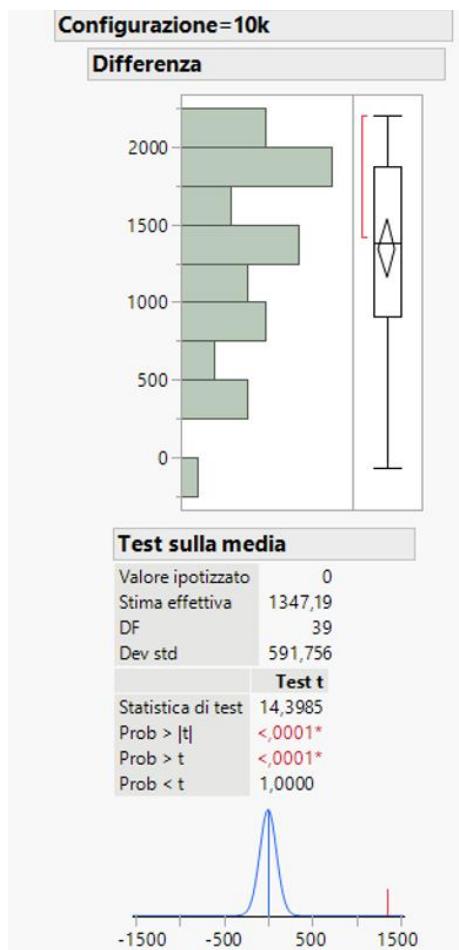
benchmark su due macchine differenti sotto le stesse condizioni (entrambe in carica, nessun altro programma in esecuzione, stessa macchina virtuale riavviata dopo ogni esecuzione...); inoltre, i parametri e le configurazioni sono identici, in modo tale da poter effettivamente confrontare le varie osservazioni, caratteristica sfruttata da questo particolare tipo di test. Possiamo calcolare la statistica di test t sfruttando la media della distribuzione della differenza tra ogni coppia di osservazioni, divisa per il rapporto tra la deviazione standard della differenza e la radice quadrata del numero di campioni totale:

$$t = \frac{\bar{x}_d}{\frac{s_d}{\sqrt{n}}}$$

Per effettuare il test è stato utilizzato JMP come già fatto in precedenza oltre a uno script Matlab che fa uso della funzione ttest: in entrambi, la differenza tra le medie dei due gruppi è significativa se il valore di p-value ottenuto è minore di 0.05, corrispondente al livello di significatività; il test dovrà inoltre essere ripetuto per tre volte, una per ogni carico a nostra disposizione. Infine, abbiamo valutato la potenza statistica dei test effettuati utilizzando un ulteriore script Matlab.

I risultati ottenuti, per ogni tipo di carico, sono mostrati nelle immagini seguenti:

1.5.1 10K corpi



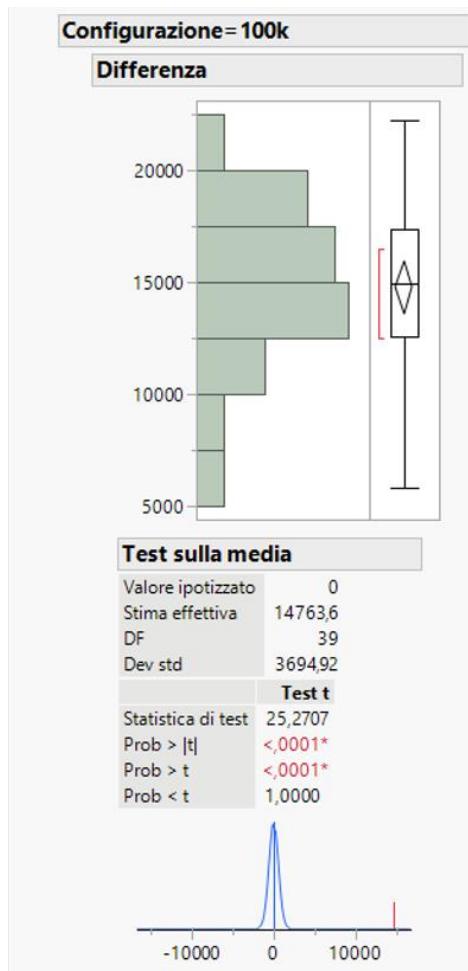
Configurazione: 10k

La differenza tra le medie è statisticamente significativa (t-test).

Valore p del t-test: 3.3942e-17

Potenza statistica del test: 1

1.5.2 100K corpi



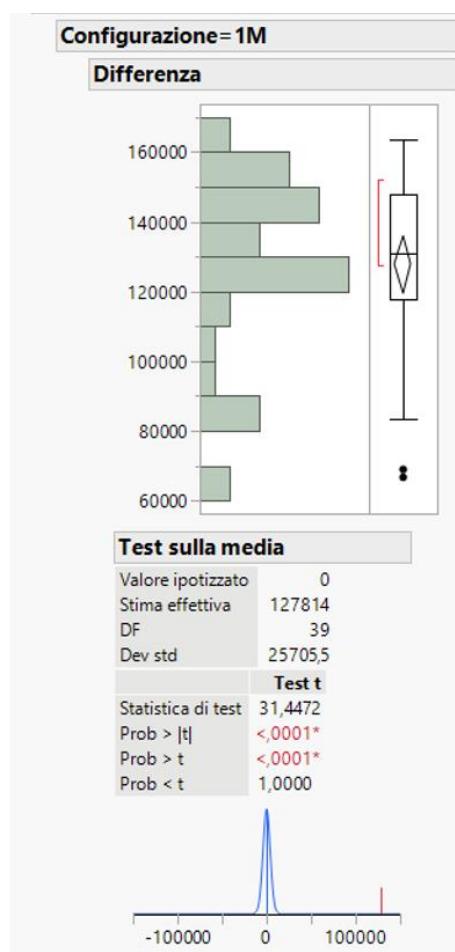
Configurazione: 100k

La differenza tra le medie è statisticamente significativa (t-test).

Valore p del t-test: 8.6596e-26

Potenza statistica del test: 1

1.5.3 1M corpi



Configurazione: 1M

La differenza tra le medie è statisticamente significativa (t-test) .

Valore p del t-test: 2.5358e-29

Potenza statistica del test: 1

Tramite questi test siamo in grado di confrontare la media osservata con un valore ipotizzato, in questo caso 0, in modo tale da determinare se la differenza tra queste due quantità sia statisticamente significativa. In particolare, il valore $prob > |t|$ è minore di 0.0001 per ognuna delle configurazioni studiate, molto inferiore al livello di significatività pari a 0.05; quindi possiamo affermare che la media della differenza studiata è significativamente diversa da zero con un livello alto di confidenza; anche il valore $prob > t$ risulta essere inferiore a 0.0001, il che significa che la media osservata è significativamente maggiore di zero, indicando una differenza positiva tra i tempi dei due sistemi. Inoltre, tramite la stima della media della distribuzione confermiamo che i risultati dei due punti precedenti sono corretti; infine, tramite Matlab verifichiamo tramite i valori di p-value la differenza di ordini di grandezza con il livello di significatività dello 0.05.

1.6 Conclusioni

Tramite l'analisi statistica e i test di ipotesi abbiamo avuto la possibilità di verificare una differenza significativa nelle prestazioni dei due sistemi non casuale. I risultati confermano quanto ipotizzato inizialmente prima dell'inizio dell'analisi, in quanto il PC Lenovo risulta avere delle specifiche migliori rispetto a quelle del PC ASUS, soprattutto per quanto riguarda il processore, di generazione più recente rispetto a quello del secondo; difatti, i risultati mostrano delle prestazioni significativamente migliori per il PC Lenovo rispetto all'ASUS.

2. PCA & Clustering

2.1 Introduzione

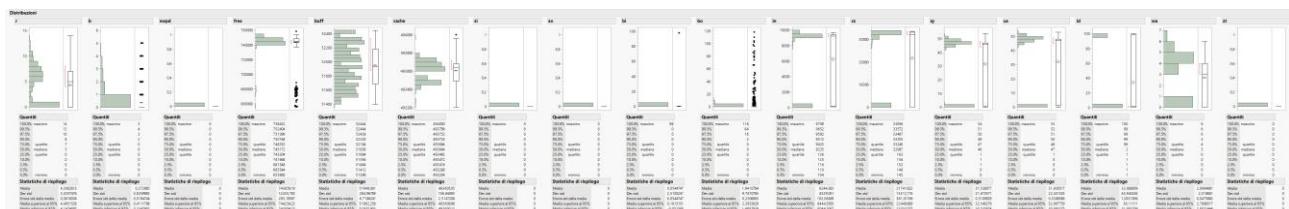
L'obiettivo di questo homework è stato **analizzare l'effetto delle operazioni di prefiltraggio, di PCA e di clustering** su un dataset di misurazioni VMStat, così da scegliere il miglior trade-off per caratterizzarlo in un workload sintetico.

Il tool utilizzato per effettuare queste operazioni è stato JMP, mentre per calcolare la devianza totale persa è stato utilizzato uno script python.

2.2 Prefiltraggio

Le prime operazioni effettuate nel dataset sono state:

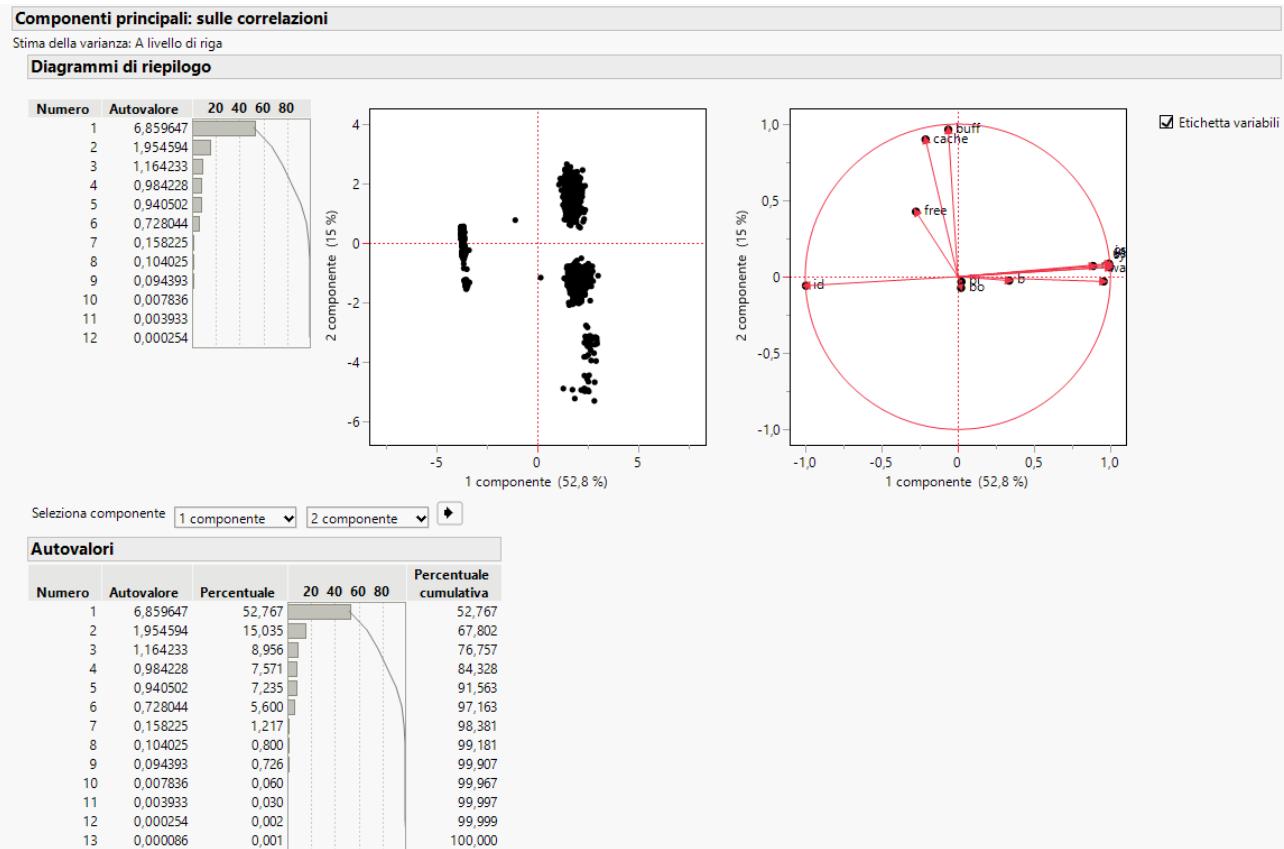
- Rimozione prima riga, in quanto corrispondeva ad una fase di boot-up del sistema
- Valutazione degli histogrammi frequentziali di tutte le colonne per valutare se statisticamente rilevanti ed analisi degli outlier



In particolare, sono state rimosse le seguenti componenti: swpd, si, so, st, bi in quanto costanti o statisticamente insignificanti.

2.3 PCA

Dopo una prima pulizia del dataset, è stata poi effettuata la **Principal Component Analysis** che permette di **portare le componenti originali del dataset in un nuovo spazio**, in cui tali componenti principali sono la somma pesata delle componenti originali e sono **incorrelate** tra di loro.



Sempre tramite JMP, sono poi stati calcolati gli autovalori della matrice di covarianza, così da poter estrarre le componenti principali in base ai rispettivi valori di percentuali cumulative.

Per questo homework sono state selezionate 2, 3, 5 e 6 componenti principali.

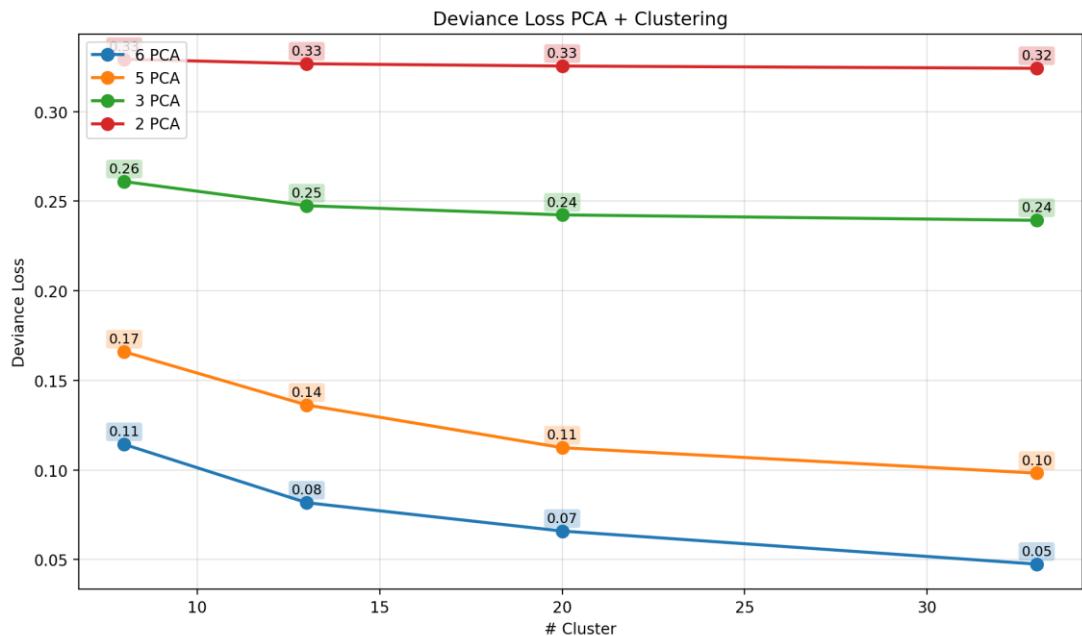
# COMPONENTI	PERCENTUALE CUMULATIVA
2	67.802
3	76.757
5	91.563
6	97.163

2.4 Clustering

È stato poi effettuato il clustering sulle componenti principali selezionando 33, 20, 13, 8 cluster per tutti e quattro i casi, utilizzando il **metodo di Ward** che permette di minimizzare la varianza intra-cluster e massimizzare quella inter-cluster.

Tramite script python è stata poi calcolata e plottata la devianza totale persa calcolata come $devianzaPersa = 1 - (devianzaPostPCA \cdot devianzaPostClustering)$, per ogni permutazione del numero di componenti principali e numero di cluster.

PCA	Cluster	total_dev_lost
2	8	0.329196
2	13	0.326651
2	20	0.325379
2	33	0.324128
3	8	0.260935
3	13	0.247463
3	20	0.242331
3	33	0.239295
5	8	0.165975
5	13	0.136263
5	20	0.112436
5	33	0.098260
6	8	0.114311
6	13	0.081759
6	20	0.065946
6	33	0.047507



Come si può osservare **con un numero ridotto di componenti principali, aumentare il numero di cluster non comporta quasi nessun miglioramento** data la grande quantità di devianza totale persa con la PCA.

Invece con un numero più elevato di componenti principali si possono notare meglio gli effetti dell'aumento del numero di cluster, permettendo di ridurre la devianza totale persa, anche se si può notare come **superati i 20 cluster la differenza in termini di devianza totale persa tende a diminuire velocemente**.

In questo caso un buon trade-off è l'utilizzo di **5 componenti principali e 20 cluster**, e a partire da questa scelta è stato ottenuto un workload sintetico scegliendo un rappresentante per ogni cluster randomicamente.

	<input checked="" type="checkbox"/> r	<input checked="" type="checkbox"/> b	<input checked="" type="checkbox"/> free	<input checked="" type="checkbox"/> buff	<input checked="" type="checkbox"/> cache	<input checked="" type="checkbox"/> bo	<input checked="" type="checkbox"/> in	<input checked="" type="checkbox"/> cs	<input checked="" type="checkbox"/> us	<input checked="" type="checkbox"/> sy	<input checked="" type="checkbox"/> id	<input checked="" type="checkbox"/> wa	<input checked="" type="checkbox"/> Cluster	
<input checked="" type="checkbox"/> Σ	1	6	0	670124	51396	493268	4	8800	29703	46	50	1	3	1
	2	6	0	656248	51420	493876	0	8220	27066	50	44	2	4	2
	3	8	2	664612	51412	493500	0	8572	28637	52	42	3	4	3
	4	6	0	745300	51620	493484	0	9347	33146	51	44	1	4	4
	5	11	1	745596	51612	493484	6	9280	33238	51	45	1	3	5
	6	4	2	741728	51540	493492	0	9530	33353	50	46	1	3	6
	7	4	3	746452	51612	493448	0	9514	33200	48	47	1	4	7
	8	3	3	749688	52320	493740	0	9054	31117	48	45	2	5	8
	9	7	0	665544	51412	493684	40	8378	27814	46	47	3	5	9
	10	6	1	745004	51580	493476	28	9389	33321	47	49	1	4	10
	11	0	0	747868	52080	493664	24	141	163	1	1	99	0	11
	12	5	0	746912	52380	493740	54	9561	33257	46	49	1	4	12
	13	0	0	745348	52008	493628	84	146	174	0	1	99	0	13
	14	9	0	747664	52304	493652	0	9320	33286	49	46	1	4	14
	15	6	0	747240	52380	493740	0	9422	33144	50	45	1	3	15
	16	7	1	743184	52264	493668	0	9480	33435	45	51	1	4	16
	17	4	2	745560	52304	493720	0	9571	33328	46	50	1	4	17
	18	7	0	656388	51420	494000	16	8441	27725	48	45	2	5	18
	19	0	0	746104	51796	493492	0	115	152	0	1	99	0	19
	20	0	0	744340	52056	493656	0	118	151	1	1	99	0	20

2.5 Clustering senza PCA

Infine, sono stati valutati i risultati del processo di clustering senza aver prima effettuato la PCA, dalla quale abbiamo ottenuto:

CLUSTER	DEVIAZIONE PERSA
33	0.035680
20	0.053068
13	0.075545
8	0.107770

Come previsto, l'utilizzo del clustering senza previa PCA permette di perdere molta meno devianza, anche se con il rischio di lavorare su colonne correlate tra di loro, avendo a disposizione in partenza il 100% della devianza del dataset.

3. Web Server

3.1 Capacity Test

3.1.1 Introduzione

Con il capacity test è stata caratterizzata la capacità di un server di rispondere alle richieste all'aumentare del carico di lavoro, per trovare due parametri fondamentali, ovvero la **Knee Capacity** e la **Usable Capacity**.

Questa valutazione viene effettuata misurando due metriche di performance di alto livello, ovvero il **Throughput** ed il **response time**. Queste metriche sono di fondamentale importanza da collezionare dato che la Knee Capacity, il punto ottimale di lavoro del server, corrisponde proprio al **miglior rapporto tra Throughput e response time**, mentre la Usable Capacity è il punto di lavoro oltre il quale il **response time cresce drasticamente**.

3.1.2 SUT

La macchina UT è una vm:

- CPU: 1 core
- RAM: 1Gb
- OS: Ubuntu 24.04 LTS
- Hypervisor: KVM

Tale VM espone alla macchina host un server web Apache 2 contenente delle risorse di diversi formati e diverse dimensioni, che variano da un centinaio di Kb a qualche Mb.

La macchina host ha le seguenti caratteristiche:

- CPU: AMD Ryzen 7 5700U
- RAM: 18 gb DDR4 3200Mhz
- OS: Arch Linux
- Apache Jmeter version: 5.6.3.

3.1.3 Test plan

Il test plan è composto da un **singolo Thread Group** con la seguente configurazione:

- **Numero di thread:** 50
- **Ramp-up period:** 25 s
- **Duration:** 300 s

Tale thread group contiene:

- **Simple Data Writer:** raccoglie i dati di alto livello

- **Summary Report:** report sintetico dell’andamento del test
- **Random Controller:** permette di randomizzare le richieste HTTP
 - **Richieste HTTP**
 - **Constant Throughput Timer:** imposta il numero di richieste al secondo

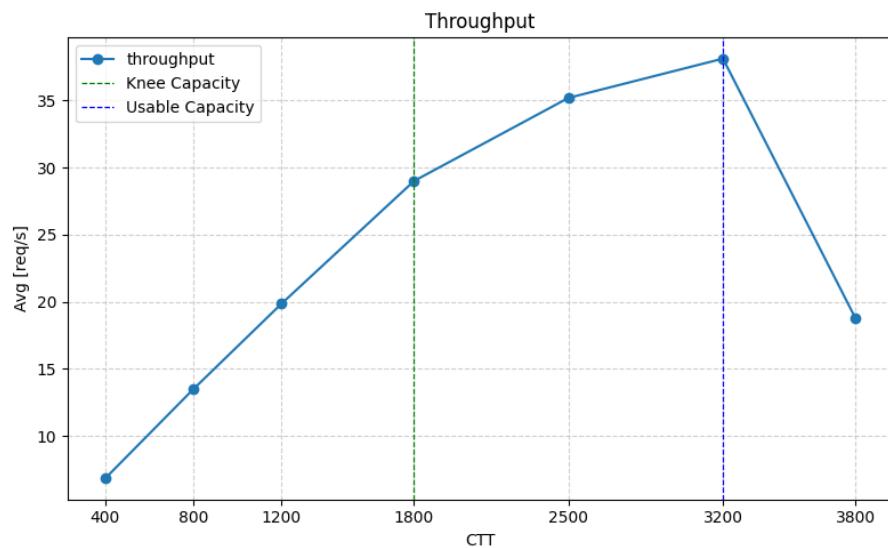
Per questo test sono state effettuate tre ripetizioni per ogni tasso del CTT, e sono stati raccolti sia i parametri di alto livello con **JMeter** che di basso livello con **VMStat**.

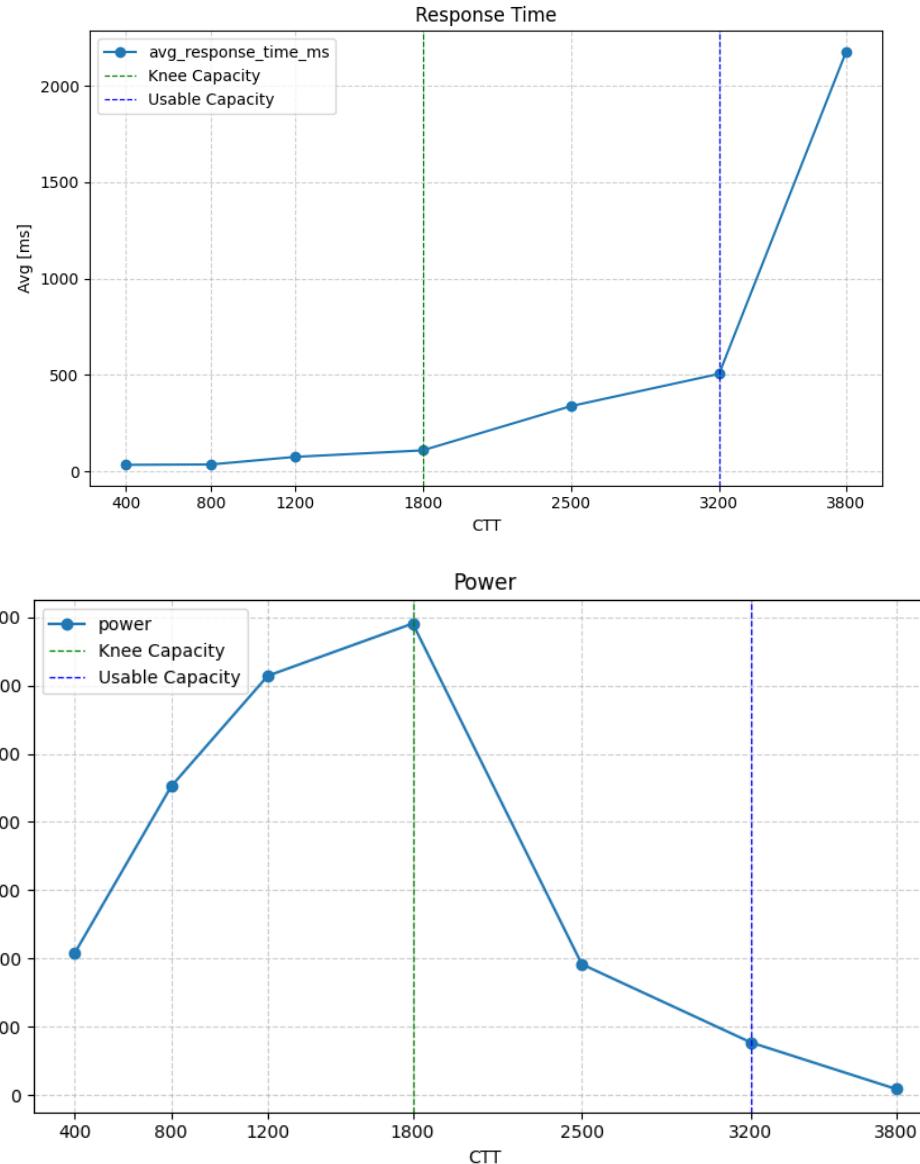
3.1.4 High Level analysis

Tramite uno script python sono stati realizzati dei plot dei valori medi per:

- throughput
- response time
- potenza ($\frac{\text{throughput}}{\text{response time}}$)

al variare del CTT, permettendo di ricavare la Knee Capacity e la Usable Capacity.





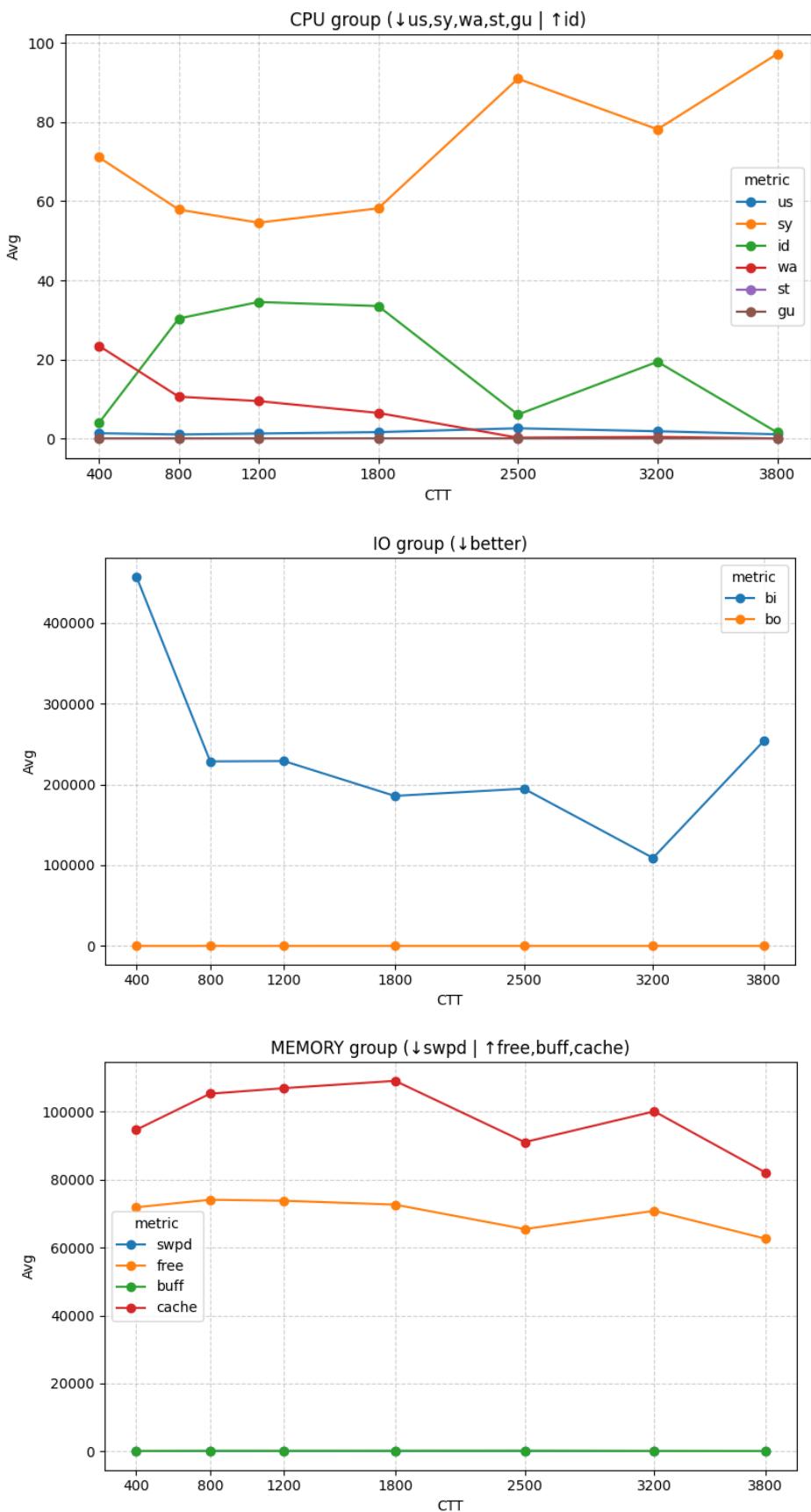
Come si può evincere dai grafici:

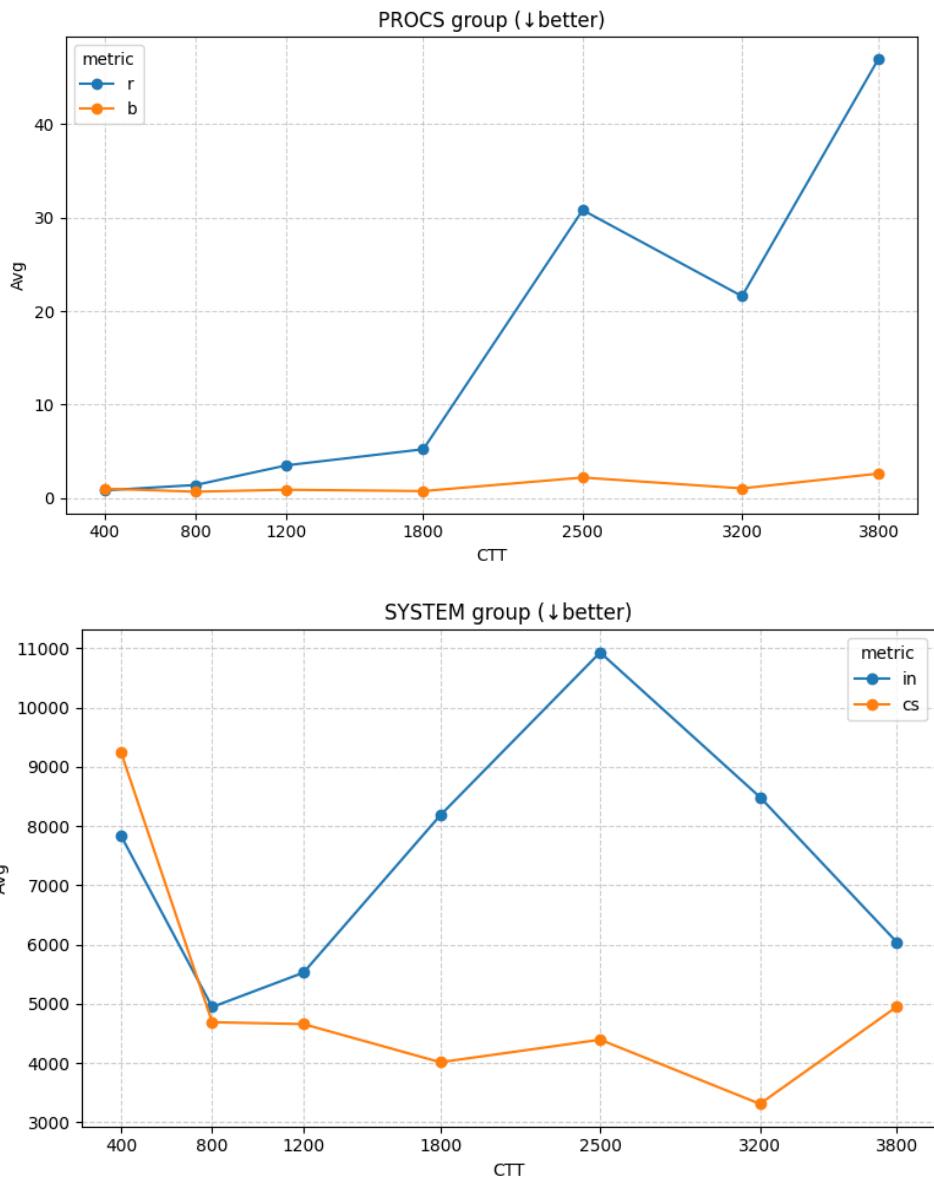
- Knee Capacity: corrisponde al **picco della potenza** ed è quindi pari a circa **1800** richieste al minuto
- Usable Capacity: corrisponde al punto oltre il quale il **response time cresce drasticamente** e corrisponde a circa **3200** richieste al minuto

Quindi il punto di lavoro ottimale di questo server è intorno alle 1800 richieste al momento

3.1.5 Low Level analysis

Analizzando le metriche di basso livello è possibile individuare eventuali **bottleneck** nel sistema, che lo portano a non poter più soddisfare le richieste che riceve.





Si può notare come con l'aumentare del carico richiesto aumenti considerevolmente il **tempo di sistema (sy) rispetto al tempo idle (id)** della CPU.

In particolare, si osserva come a partire da 2500 CTT, il **system time (sy)** raggiunga il 90% dell'utilizzo della CPU, fino ad arrivare al 97% al carico massimo, mentre l'**idle time (id)** crolla dall'33% al solo 1.6%.

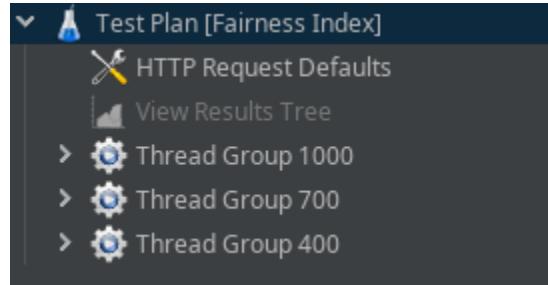
La coda di processi in attesa (r) cresce drasticamente **da pochi processi a circa 50**, evidenziando una grave congestione della CPU.

Il tempo CPU utente (us) rimane invece costantemente basso (1-2%), indicando che **il bottleneck è nelle operazioni del kernel e non nel codice applicativo**.

Quindi la **bottleneck del sistema è proprio la CPU**.

3.1.5 Fairness Index

Per la valutazione del **fairness index** è stato create un nuovo test plan contenente tre thread group differenti che fanno richieste concorrentemente per delle risorse differenti, utilizzando CTT differenti. Il test della durata di cinque minuti è stato ripetuto per tre volte, così da poter poi utilizzare il throughput medio delle ripetizioni.



Tramite script python è stato poi calcolato il fairness index come $f(x_1, \dots, x_n) = \frac{(\sum_{i=1}^n x_i)^2}{n \sum_{i=1}^n x_i^2}$, dove x_i è il normalized throughput $x_i = \frac{\text{throughputMeasured}}{\text{throughputExpected}}$

Di seguito i risultati:

THREAD GROUP	MEASURED THROUGHPUT	NOMINAL THROUGHPUT	NORMALIZED THROUGHPUT
1	16.305	16.667	0.978
2	6.661	6.667	0.999
3	11.486	11.667	0.984

Ottenendo un **fairness index pari a 0.999**.

3.1 Workload Characterization

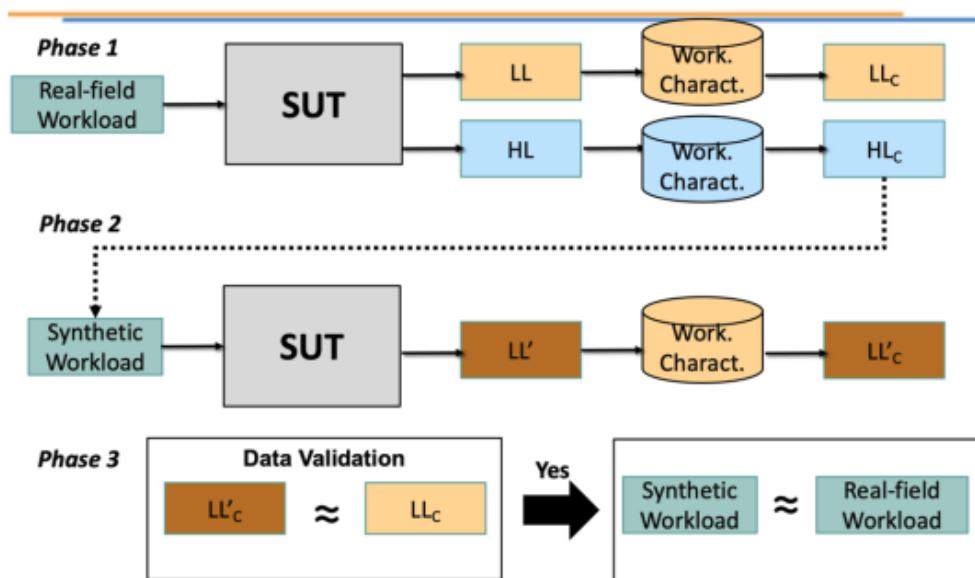
3.1.1 Introduzione

Per caratterizzazione di un workload si intende il processo di sintesi che consente di rappresentarlo attraverso un **set di attributi capaci di spiegare un'alta percentuale della devianza originale, ma con una dimensionalità ridotta**.

Un **workload sintetico** è, quindi, un **modello statistico rappresentativo del carico reale** che permette di replicarne gli effetti sul sistema utilizzando però un dataset più compatto, riducendo drasticamente tempi e costi di testing.

Tale processo può essere diviso in tre fasi:

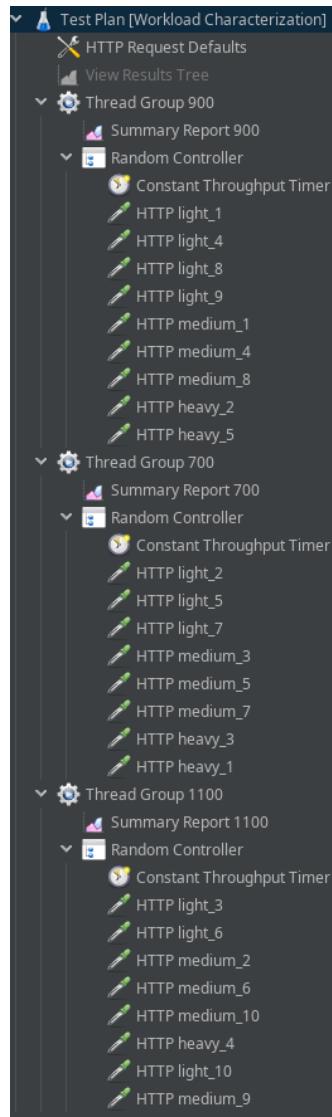
1. **Applicazione del workload reale al SUT**, collezionando parametri di alto livello e di basso livello. A partire dai dati collezioni si effettua la caratterizzazione di questi parametri tramite operazioni di prefiltraggio, PCA e clustering, così da ottenere il workload sintetico.
2. **Applicazione del workload sintetico al SUT**, collezionando i parametri di basso livello per effettuare la caratterizzazione anche di questi ultimi, utilizzando le stesse identiche operazioni applicate ai parametri di basso livello nella fase precedente.
3. **Validazione parametri di basso livello**, confrontando i parametri di basso livello caratterizzati in entrambe le fasi, tramite hypothesis testing per valutare l'equivalenza statistica.



3.1.2 Setup

- Web Server: Apache HTTP Server 2
- VM: Ubuntu 24 LTS, 1 core, 1 GB RAM
- Host: Arch Linux, AMD Ryzen 7 5700U, RAM 18 gb DDR4 3200Mhz

- JMeter: 5.6.3, tre Thread Group (CTT 900, 700, 1100) che fanno richieste HTTP a 25 risorse differenti (ogni thread group richiede delle risorse univoche, e ogni thread group viene eseguito sequenzialmente)



3.1.3 Fase 1: caratterizzazione workload reale

Dopo la raccolta dei parametri HL e LL, è stata effettuata la caratterizzazione HL_c tramite pre-filtraggio dei dati, PCA e clustering.

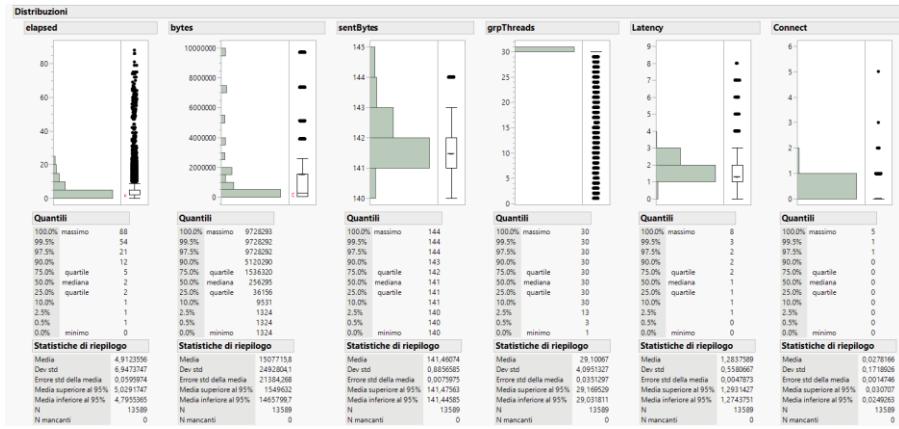
HL_c

1. Prefiltraggio dati

Rimossa la prima riga in quanto corrispondente alla fase di avvio del server.

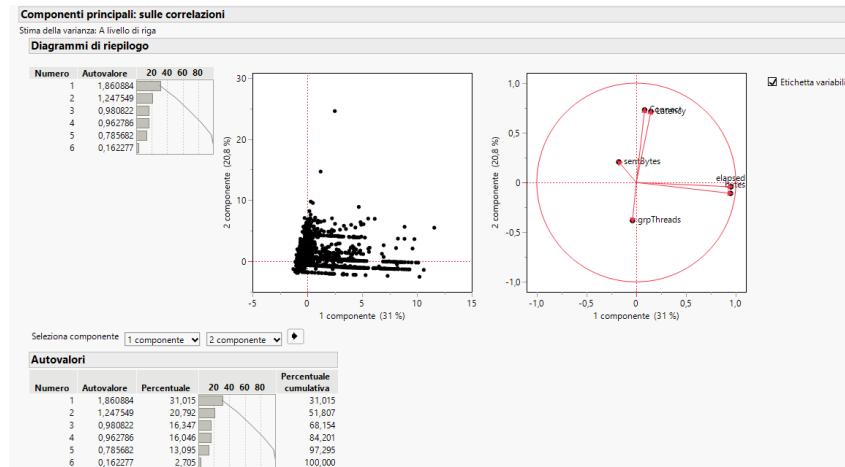
Rimosse le componenti: `timeStamp`, `responseCode`, `responeMessage`, `threadName`, `dataType`, `success`, `failureMessage`, `URL`, essendo a varianza nulla o contenenti valori categorici.

Rimossa la componente `allThreads`, essendo identica a `grpThreads`.



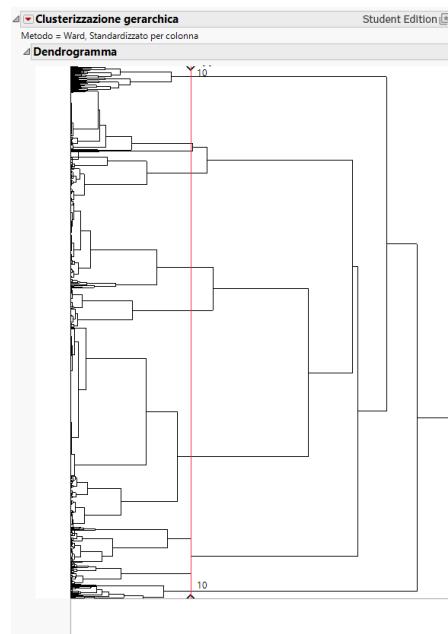
2. PCA

Effettuando il test di Bartlett e la PCA, si è scelto di utilizzare cinque componenti principali



3. Clustering

Si è scelto di utilizzare dieci cluster



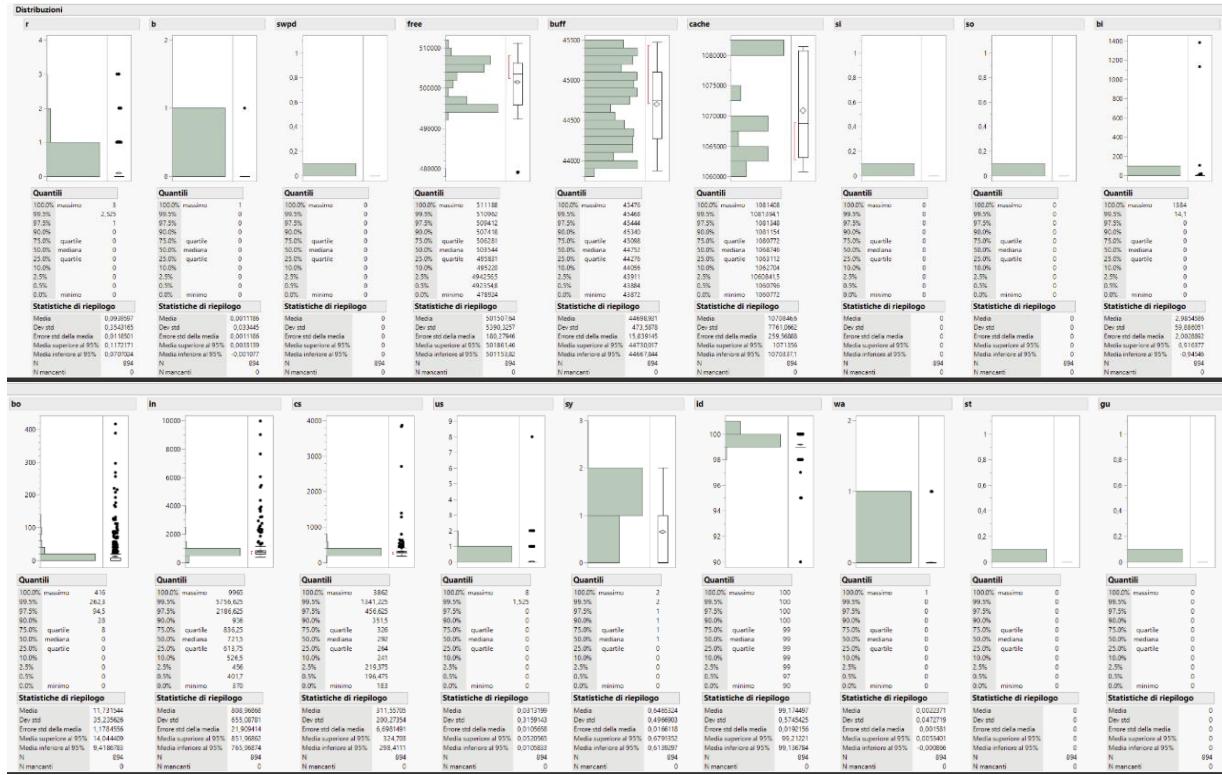
Infine, tramite script python è stata calcolata la devianza totale persa, risultata pari al 24.37%

LL_c

1. Prefiltraggio dati

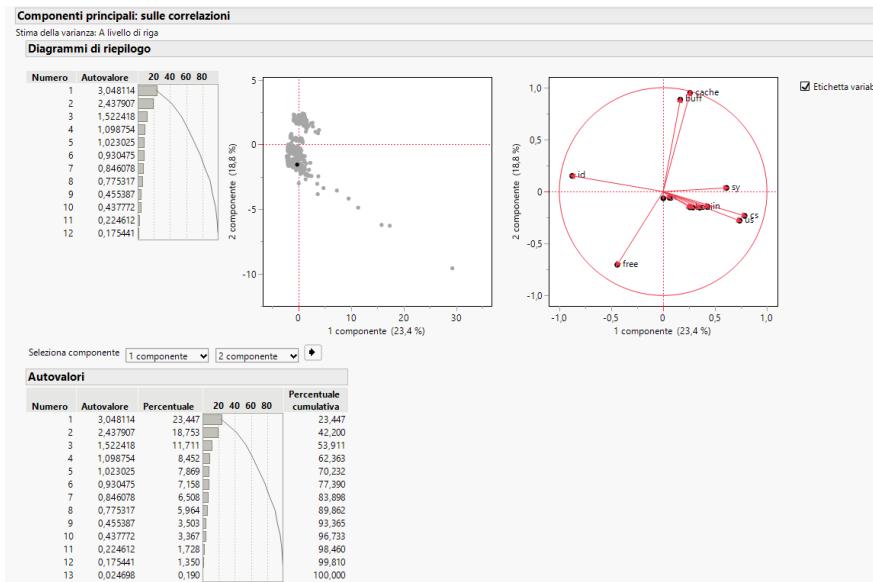
Rimossa la prima riga in quanto corrispondente alla ‘fase di avvio del server’.

Rimosse le componenti: *swpd*, *si*, *so*, *st*, *gu* essendo a varianza nulla.



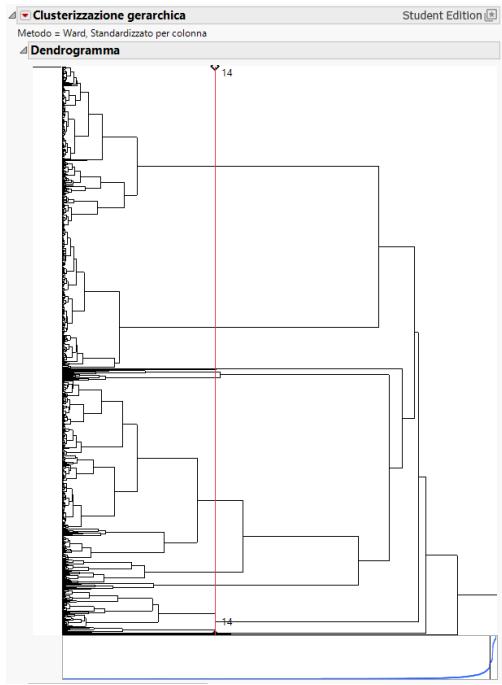
2. PCA

Effettuando il test di Bartlett e la PCA, si è scelto di utilizzare otto componenti principali



3. Clustering

Sono poi stati scelti 14 cluster



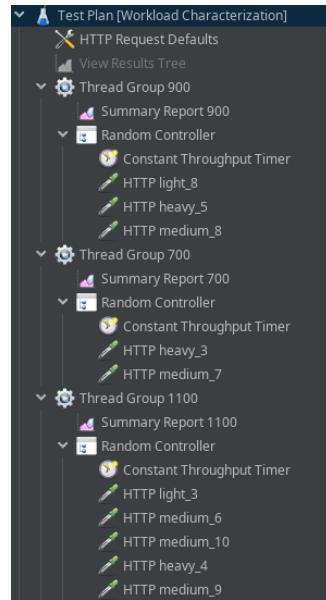
Infine, è stata calcolata la devianza totale persa tramite script python, pari al 24.36%.

3.1.4 Fase 2: LL' e LL'_c

Una volta caratterizzato HL, è stata **selezionata una richiesta per ogni cluster scegliendo quella effettuata più volte, ottenendo così il workload sintetico**.

Successivamente è stato creato il nuovo test plan su JMeter contenente le dieci richieste (una per ogni cluster), così da poter **effettuare il test con il workload sintetico, e misurare nuovamente le metriche di basso livello ottenendo LL'**.

Una volta ottenuto LL' , è stato caratterizzato effettuando le stesse identiche operazioni effettuate su LL , ottenendo LL'_c .

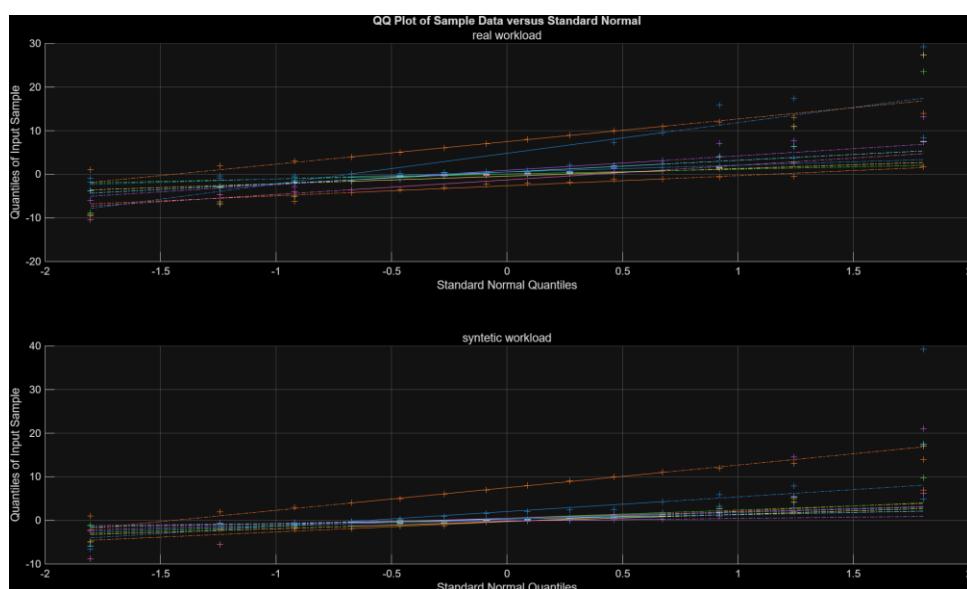


3.1.5 Fase 3: Data validation

Come ultimo step, bisogna **validare il workload sintetico**, verificando l'equivalenza statistica sui parametri di basso livello, così da valutare se gli effetti del workload sintetico sul server sono statisticamente equivalenti all'effetto del workload reale.

Come prima verifica, tramite Matlab, è stata **controllata la normalità dei dati** di entrambi LL_c e LL'_c individualmente (essendo i dati non accoppiati), visivamente tramite qq-plot e test di Kolmogorov-Smirnov:

- H_0 : i dati nel vettore provengono da una distribuzione normale standard
- H_1 : i dati nel vettore non provengono da una distribuzione normale standard



```
Kolmogorov-Smirnov test results:  
- real workload null hypothesis: 1  
- syntetic workload null hypothesis: 1
```

Data la non normalità dei dati, è stato necessario utilizzare il **test non parametrico** di Wilcoxon rank sum (con un livello di confidenza del 95%):

- H_0 : i dati sono samples provenienti da distribuzioni continue con le stesse mediane
- H_1 : i dati sono samples provenienti da distribuzioni continue con mediane differenti

```
h_wilc [0,0,0,0,0,0,0]
```

Dato che l'ipotesi nulla non viene rigettata per nessuna delle componenti principali, è possibile affermare **con un livello di confidenza del 95% che c'è equivalenza statistica tra il workload reale ed il workload sintetico.**

4. Regressione

Introduzione

In questo homework sono stati utilizzati diversi dataset di differente tipologia per realizzare dei modelli regressivi lineari.

In particolare, il modus operandi per trovare i parametri della retta di regressione è stato il seguente:

1. Verifica delle ipotesi:

- 1.1. Relazione lineare tra la variabile di risposta rispetto al predittore
- 1.2. Controllo normalità residui (test visivi e Anderson-Darling)
- 1.3. Controllo omoschedasticità residui e che non presentino trend

2. Calcolo parametri della retta di regressione:

- SE tutte le ipotesi verificate: regressione ai minimi quadrati
- SE residui non normali: test di Mann-Kendall per individuare trend e calcolo dei parametri regressivi tramite Theil-Sen

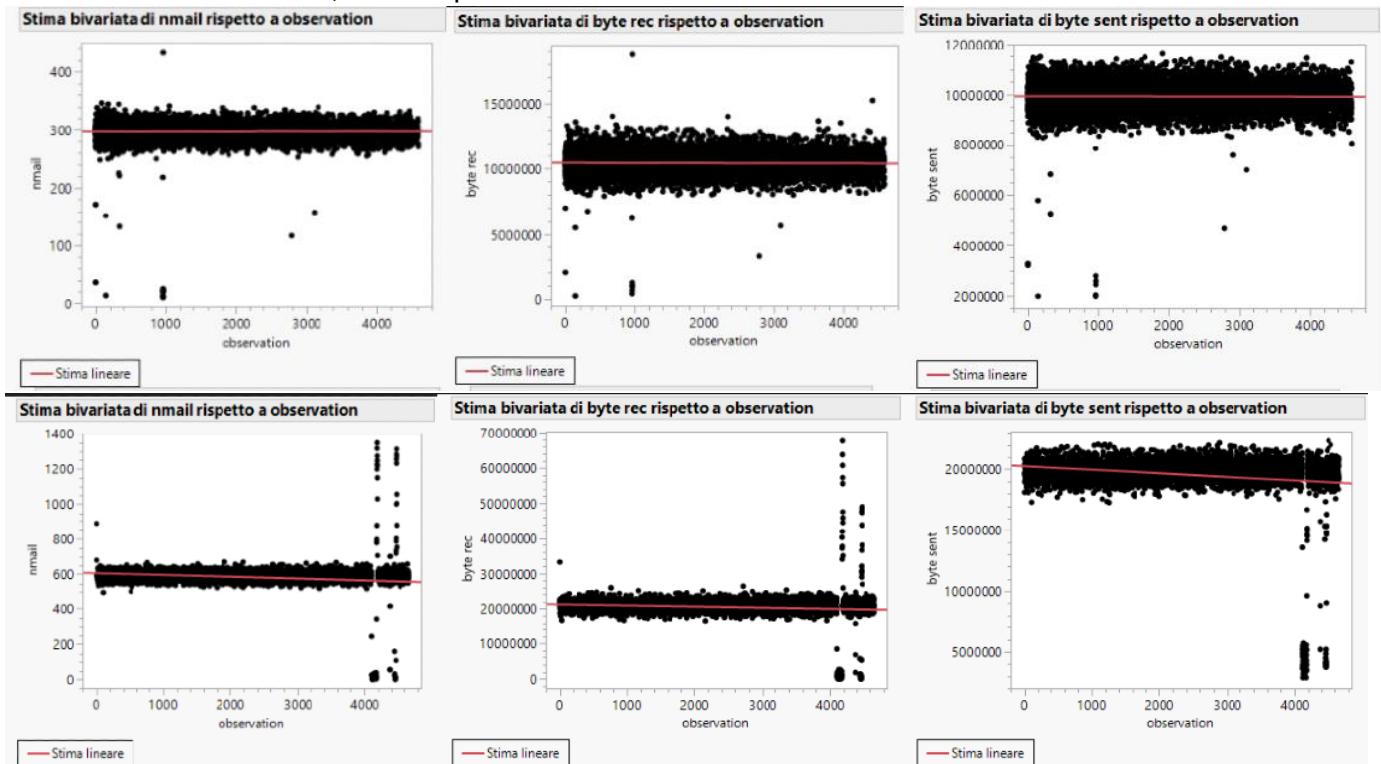
Per effettuare queste operazioni, è stato utilizzato JMP per creare i plot per verificare le assunzioni visivamente, ed eventualmente effettuare il test di Mann-Kendall.

Mentre per calcolare la stima della pendenza e dell'intercetta con Theil-Sen o effettuare le predizioni una volta stimati i parametri, sono stati utilizzati degli script Python.

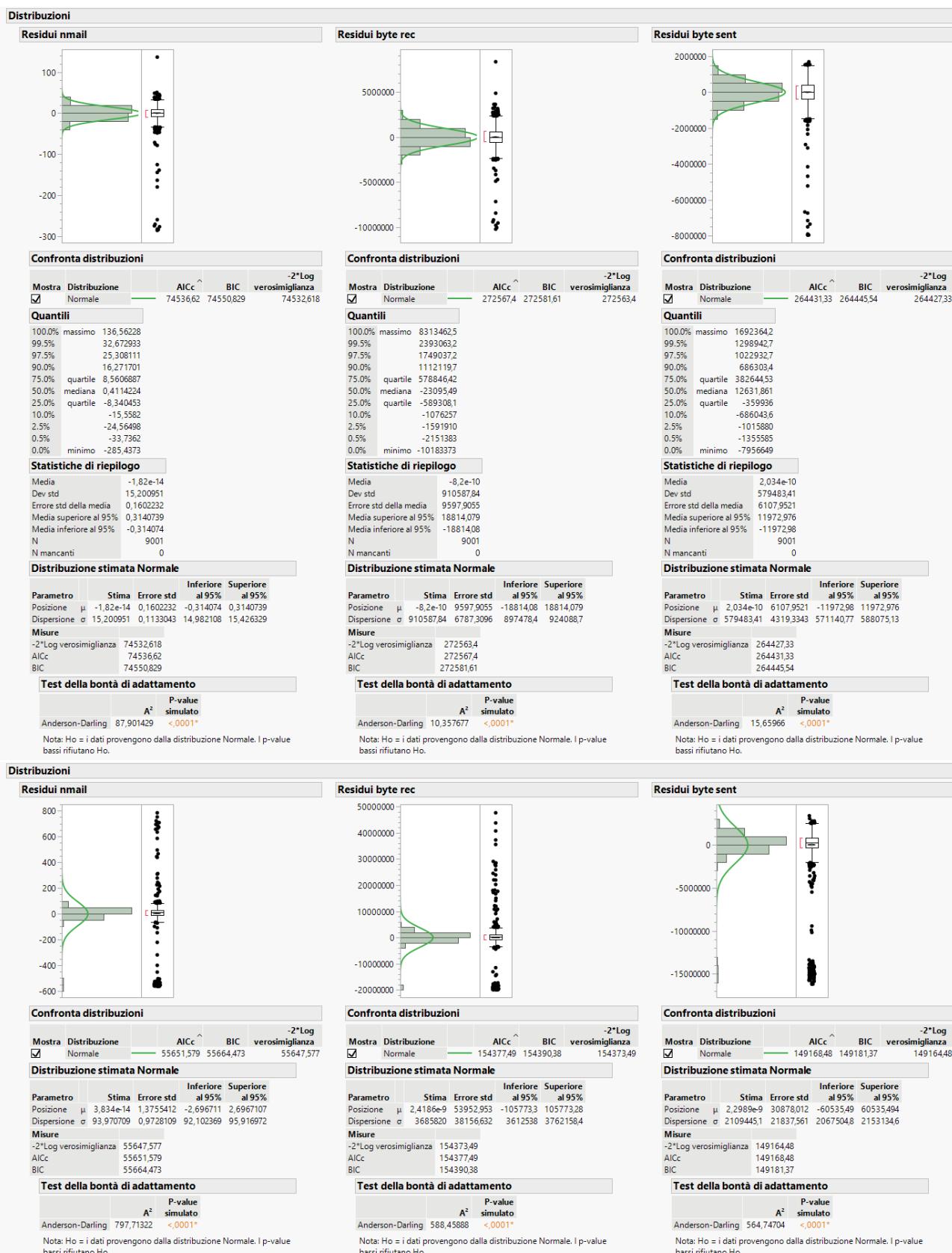
1. EXP

Rilevare e stimare eventuali trend su ognuna delle tre variabili *nmail*, *byte rec* e *byte sent*, utilizzando modelli regressivi lineari semplici, parametrici e/o non parametrici.

Per prima cosa è stato valutato visivamente la relazione lineare tra le tre variabili rispetto al numero di osservazione, così da poter anche ottenere i residui.

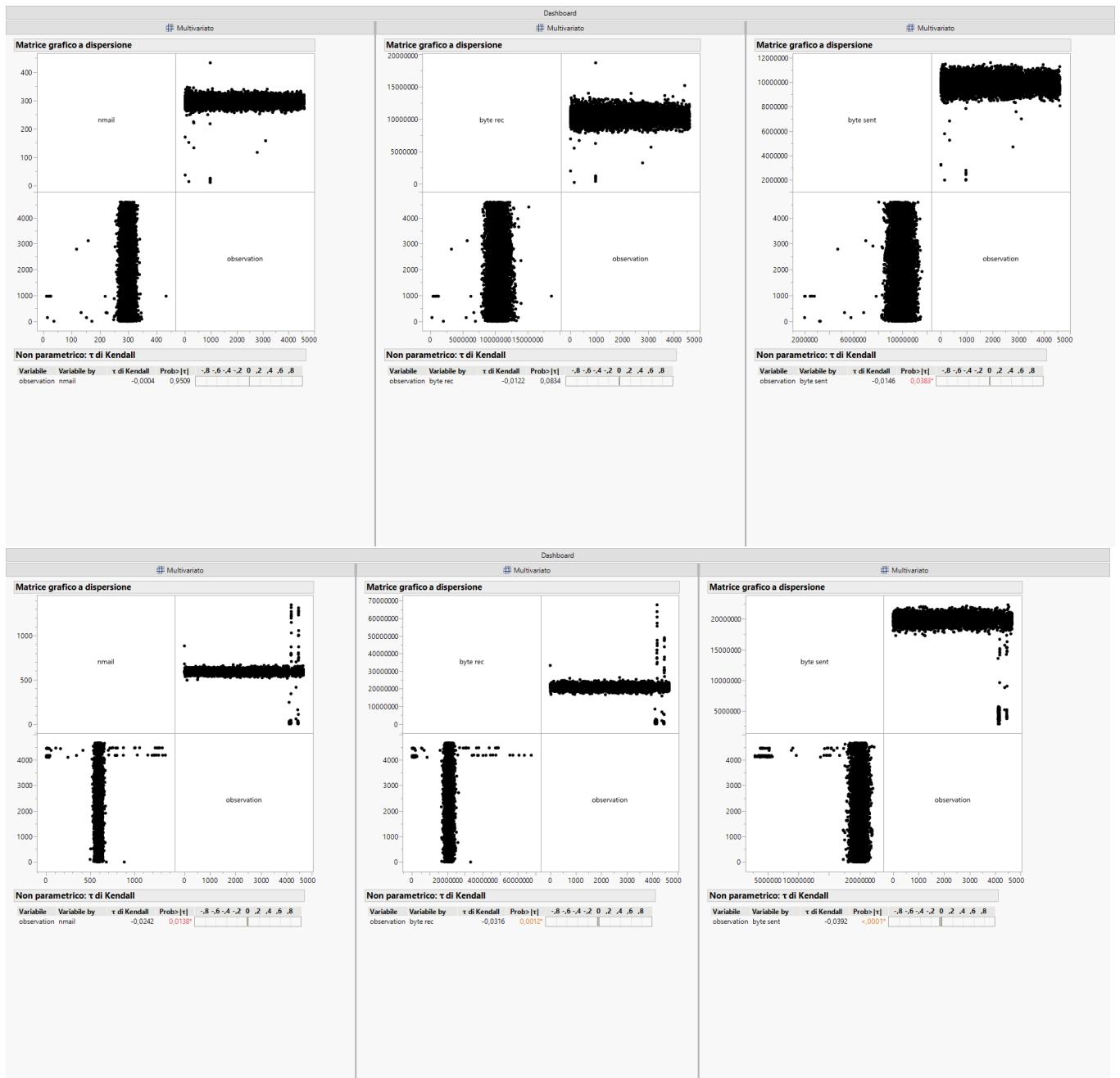


Successivamente è stata verificata la normalità della distribuzione dei residui, sia visivamente che usando il test di Anderson-Darling (alternativa al test di Shapiro-Wilk utilizzata da JMP quando $n > 2000$), con il quale è stato possibile osservare che tutti i residui sono non-normali.



Nel test di Mann-Kendall l'ipotesi nulla è che non esiste un trend, in genere si considera che esiste quando il p-value è < 0.05, inoltre il coefficiente τ compreso tra -1 e 1, indica quanto sia forte l'associazione tra le due variabili (in positivo o in negativo).

In questo caso è stato individuato un trend nel parametro byte_sent di EXP1, e in tutti i parametri di EXP2.



Una volta individuata la presenza di trend, sono stati calcolati i parametri della retta di regressione utilizzando lo stimatore di Theil-Sen.

I risultati principali sono:

- EXP1
 - Nmail e byte_rec in EXP1 hanno lo zero nell'intervallo, confermando che non esiste un trend
 - Byte_sent ha un trend monotono decrescente
- EXP2
 - Nmail è un caso limite in cui il test di Mann-Kendall rileva un trend, ma l'intervallo comprende lo zero, indice che il trend esiste ma è molto debole
 - Byte_res e byte_sent hanno un trend monotono decrescente

Sheet	Metric	Slope	Interval_Low	Interval_Up	Intercept	Trend
EXP1	nmail	0.000000	0.000000	0.000000	297.00	Non significativo
EXP1	byte rec	-12.876245	-27.426881	1.691755	10411056.14	Non significativo
EXP1	byte sent	-9.667414	-18.762445	-0.516735	9954543.87	Decrescente
EXP2	nmail	-0.000597	-0.001125	0.000000	592.39	descrescente debole
EXP2	byte rec	-48.293610	-77.547733	-18.993147	20890131.99	Decrescente
EXP2	byte sent	-34.063052	-50.672673	-17.374088	19903064.10	Decrescente

2. OS

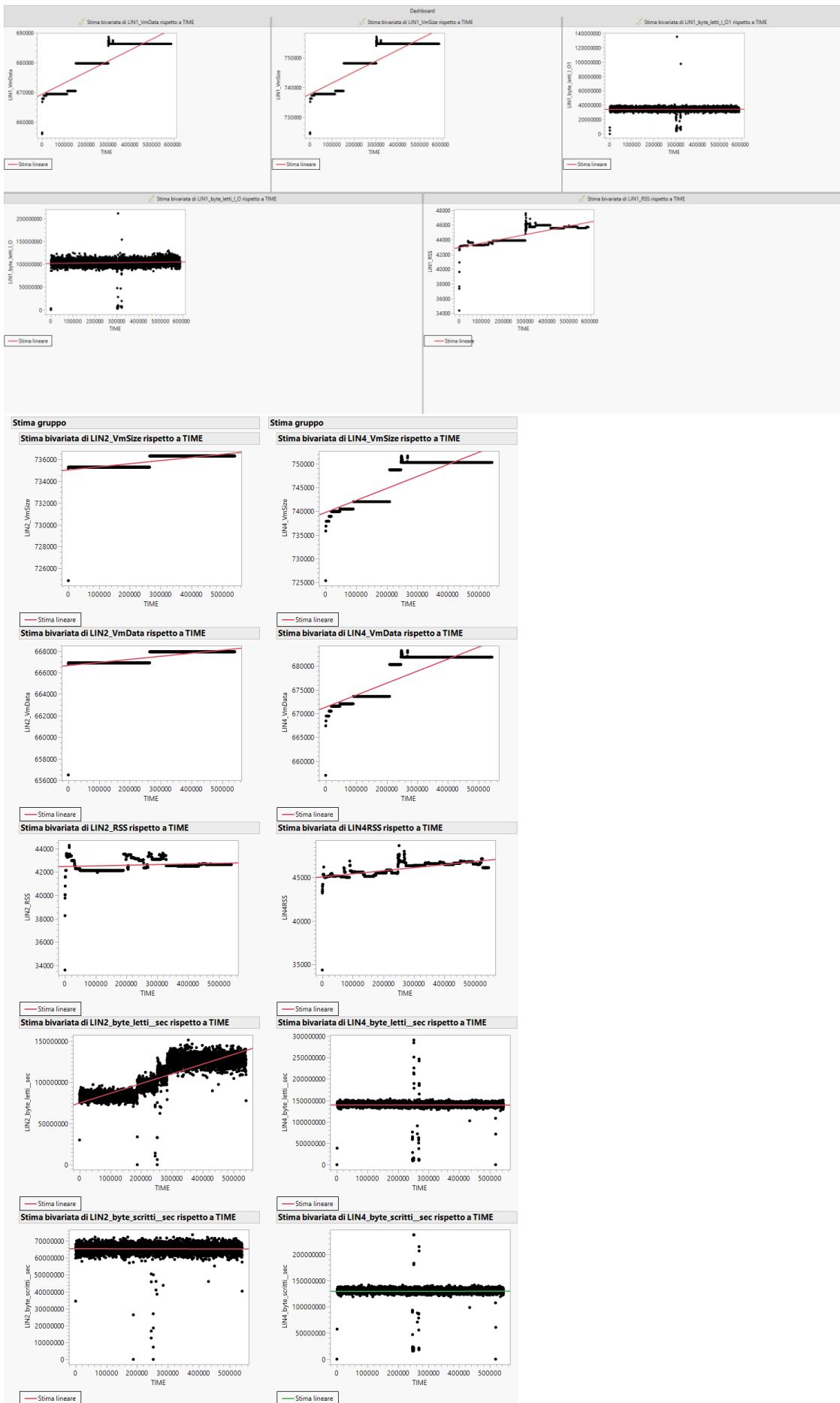
Rilevare e stimare eventuali trend sulle 5 variabili utilizzando modelli regressivi lineari semplici, parametrici e/o non parametrici.

Farlo per i tre dataset os1, os2 e os3.

Confrontare i trend individuati nei tre dataset.

Il procedimento è stato praticamente identico al caso precedente, dove è solo stato fatto un confronto finale tra i risultati dei diversi dataset.

Per prima cosa è stata valutata la relazione lineare delle variabili rispetto al tempo.

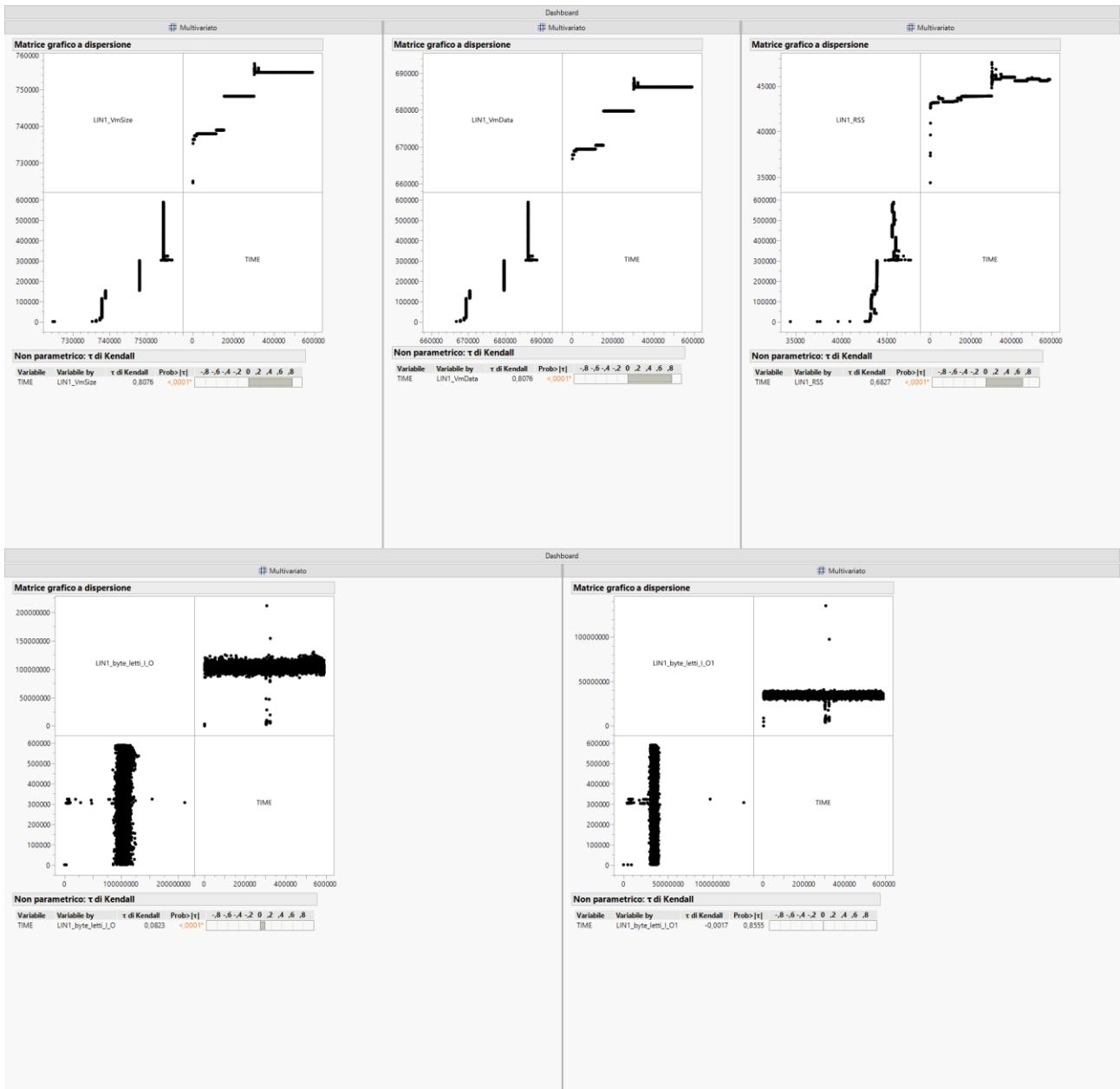


Una volta calcolati i residui è stata verificata la loro normalità

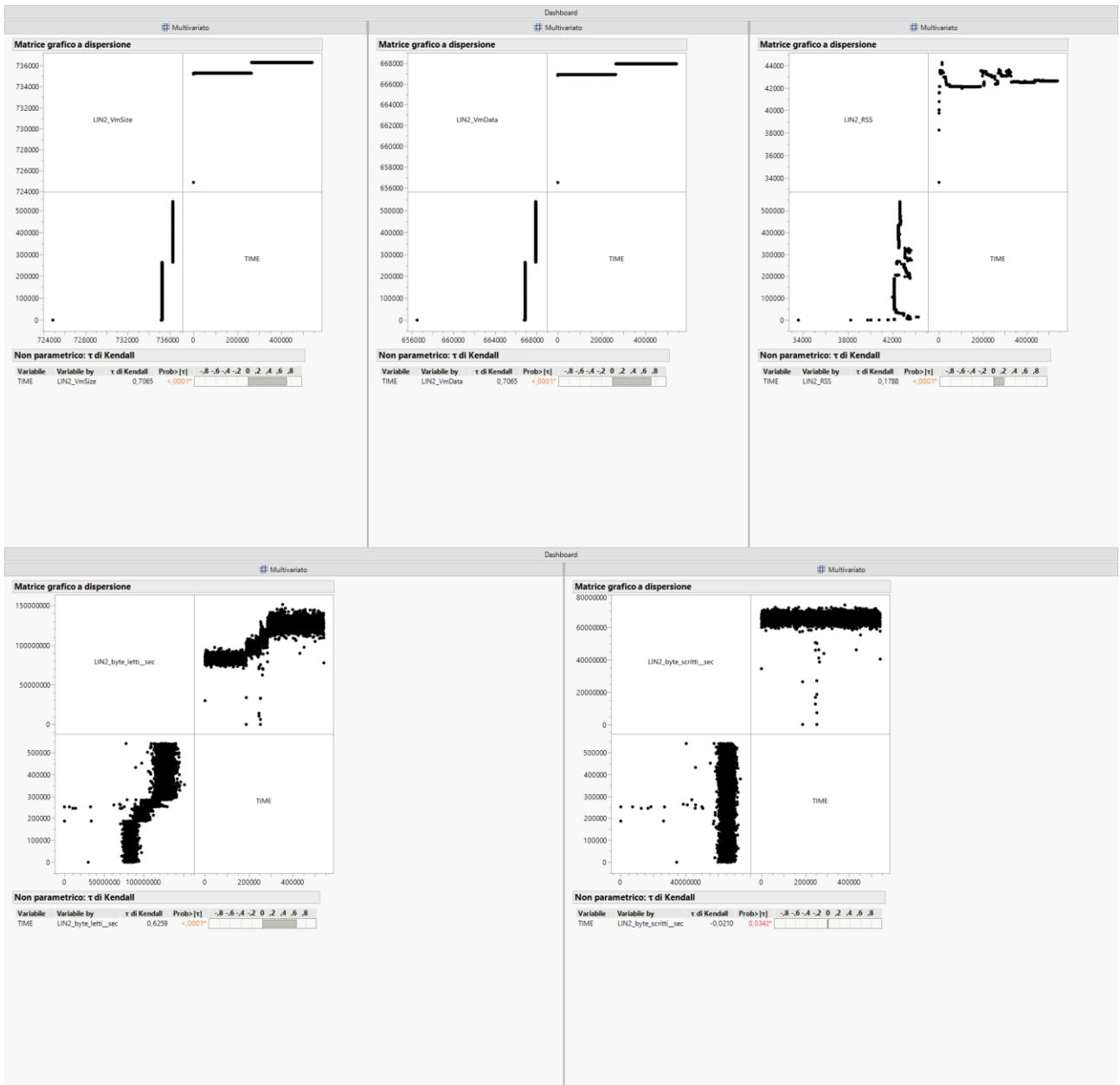


Dopo aver verificato la non normalità dei dati, è stato effettuato il test di Mann-Kendall:

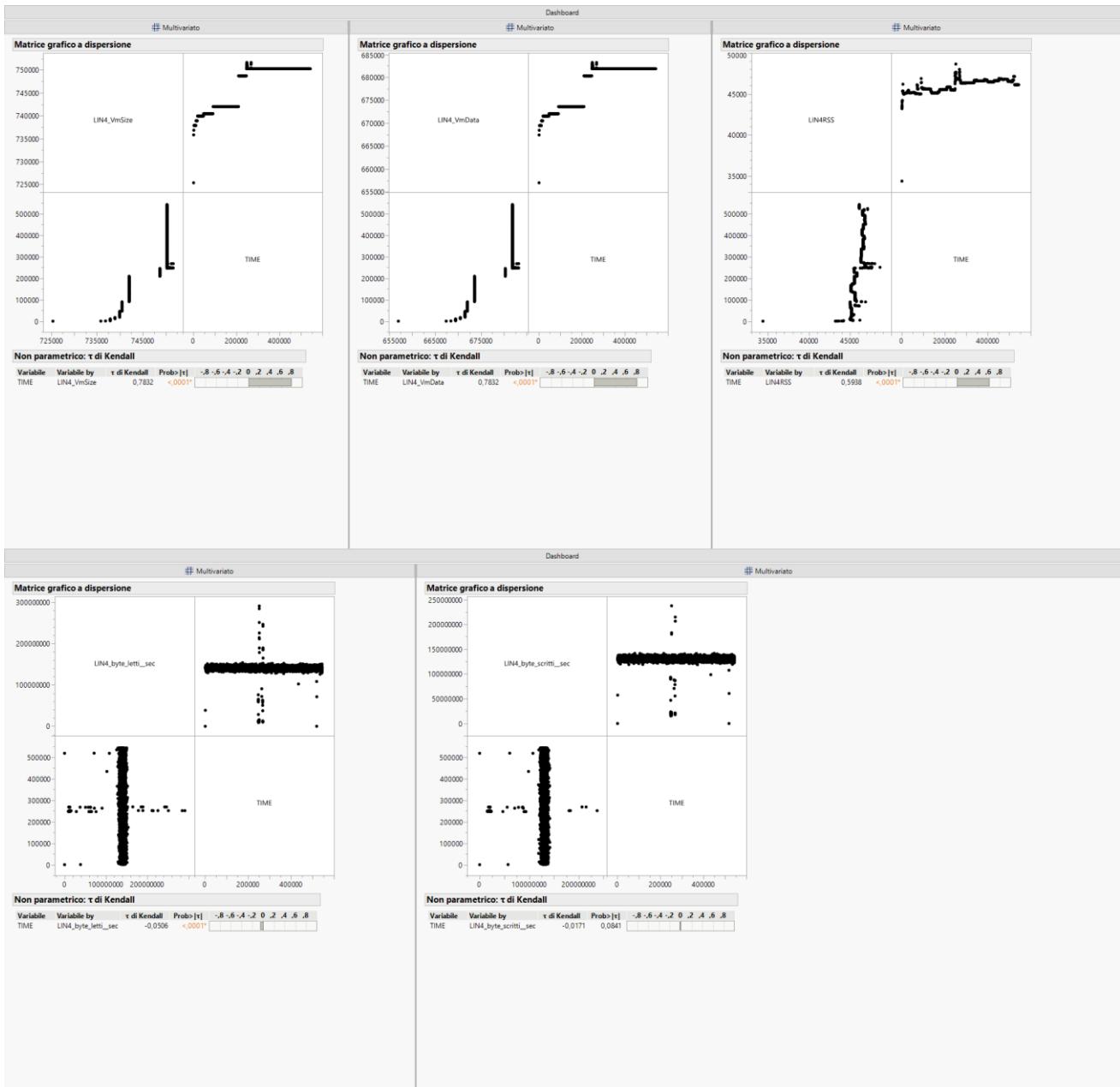
In os1, il campo byte_letti_I_O1 non ha trend.



In os2, è stato rilevato un trend in tutti i campi.



In os3, il campo byte_letti_I_O1 non ha trend.



Una volta effettuato il test di Mann-Kendall sono stati calcolati i parametri della retta di regressione con Theil-Sen.

I risultati principali sono:

- Casi limite:
 - In os2 per i campi LIN2_VmData, LIN2_VmSize, Mann-Kendall rileva un trend monotono con Tau = 0.706, Theil-Sen conferma il trend positivo ma l'intervallo tocca lo zero come limite inferiore
- Confronti per campo:
 - VmSize e VmData: os1 > os3 > os2 (os1 ha il trend crescente più marcato)
 - RSS: os1 > os3 > os2 (tutti crescenti)
 - byte_letti: os2 ha il trend più alto (114.55 byte/sec!) mentre os3 ha trend decrescente
 - byte_scritti: os2 è l'unico significativo con un trend decrescente (supponendo che il corrispettivo campo in os1 abbia solo un errore di battitura del nome)

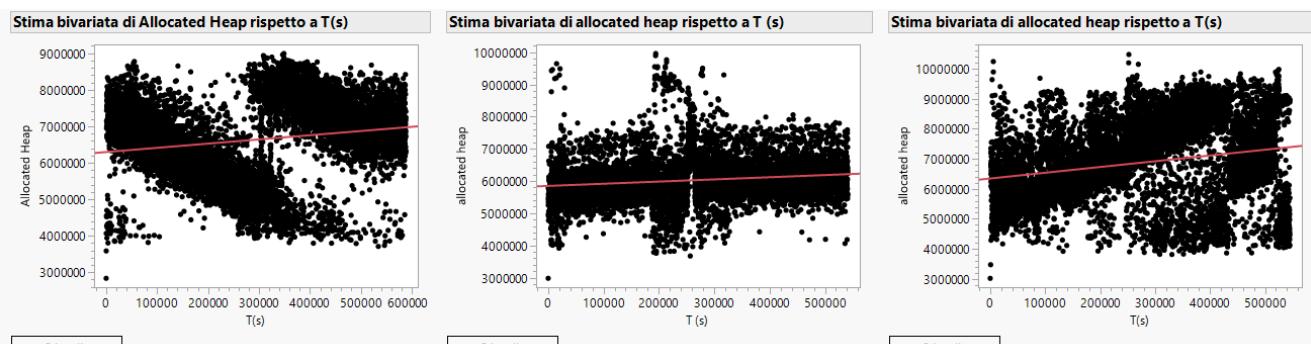
Sheet	Metric	Slope	Interval_Low	Interval_Up	Intercept	Trend
os1	LIN1_VmSize	0.031572	0.030716	0.032321	738854.39	Crescente
os1	LIN1_VmData	0.031572	0.030716	0.032321	670450.39	Crescente
os1	LIN1_RSS	0.004971	0.004877	0.005056	42428.81	Crescente
os1	LIN1_byte_letti_I_O	4.737578	3.673916	5.804599	102763229.50	Crescente
os1	LIN1_byte_letti_I_O1	-0.030814	-0.360880	0.298425	34500875.47	Non significativo
os2	LIN2_VmSize	0.001913	0.000000	0.002085	735790.38	Crescente debole
os2	LIN2_VmData	0.001913	0.000000	0.002085	667414.38	Crescente debole
os2	LIN2_RSS	0.000632	0.000556	0.000745	42384.85	Crescente
os2	LIN2_byte_letti_sec	114.553510	112.733593	116.376598	77733384.24	Crescente
os2	LIN2_byte_scritti_sec	-0.488613	-0.939001	-0.036319	65693615.58	Decrescente
os3	LIN4_VmSize	0.021659	0.021274	0.022026	744374.60	Crescente
os3	LIN4_VmData	0.021659	0.021274	0.022026	675970.60	Crescente
os3	LIN4RSS	0.003458	0.003388	0.003522	45386.47	Crescente
os3	LIN4_byte_letti_sec	-1.891049	-2.617642	-1.167760	141291080.77	Decrescente
os3	LIN4_byte_scritti_sec	-0.562464	-1.197505	0.075960	130813214.83	Non significativo

3. Heap

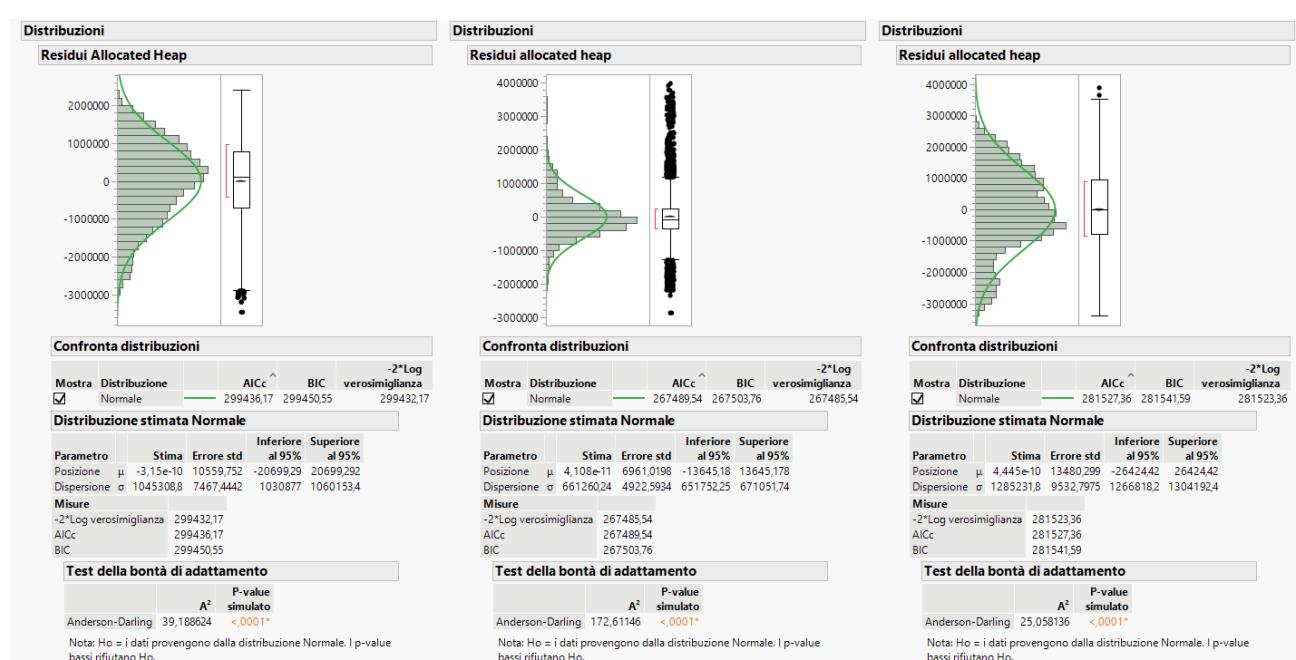
Supponendo di avere un limite massimo alla memoria heap di 1 GByte. Rilevare un eventuale trend di consumo dello heap nell'esperimento in figura. Se rilevato il trend, stimare il tempo in cui lo heap satura (failure prediction).

Anche in questo caso il procedimento è analogo ad i precedenti, l'unica differenza è che una volta trovati i parametri della retta di regressione (se esiste un trend) è stata effettuata la failure prediction.

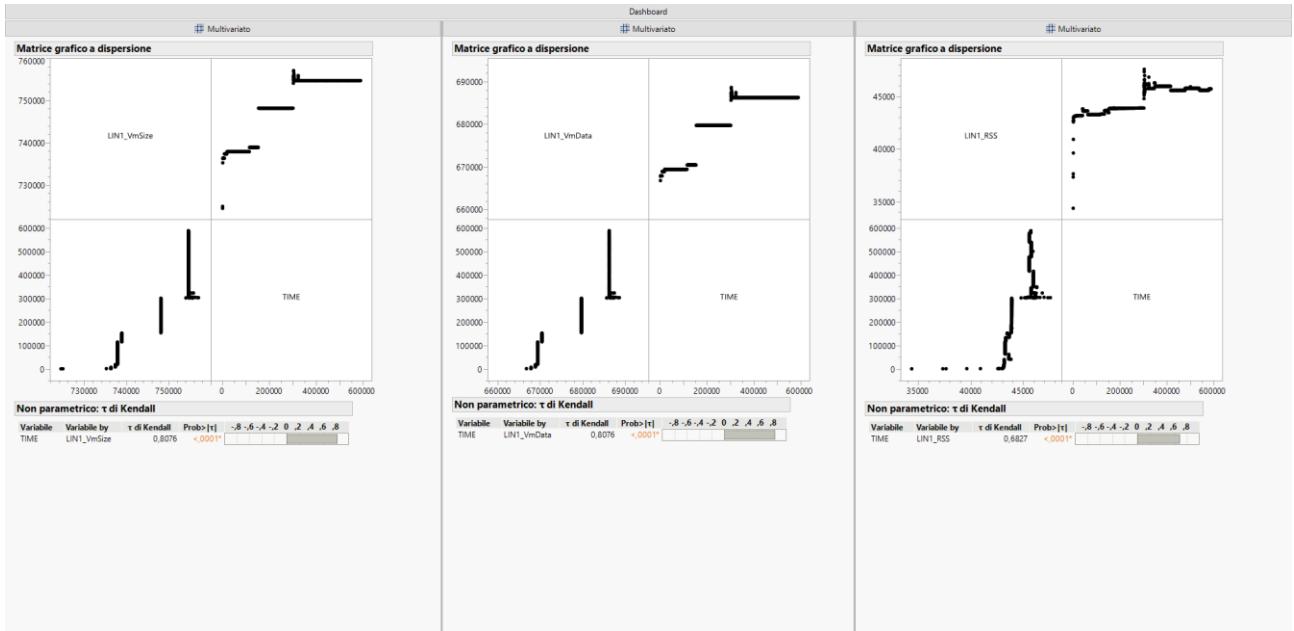
Innanzitutto, è stata valutata la relazione lineare tra l'heap allocato ed il tempo per tutte le VM, permettendo di calcolare i residui e valutare l'adjusted R², che è risultato essere relativamente basso in tutti i casi (tra 0,024 e 0,05)



Successivamente è stata verificata la normalità dei residui, dove in nessuno dei tre casi il test ha avuto successo



È stato poi effettuato il test di Mann-Kendall per individuare un trend nei dati, dove si può notare che il p-value è molto piccolo ed il coefficiente τ molto vicino a +1



Per stimare i parametri della retta regressiva è stato infine utilizzato Theil-Sen

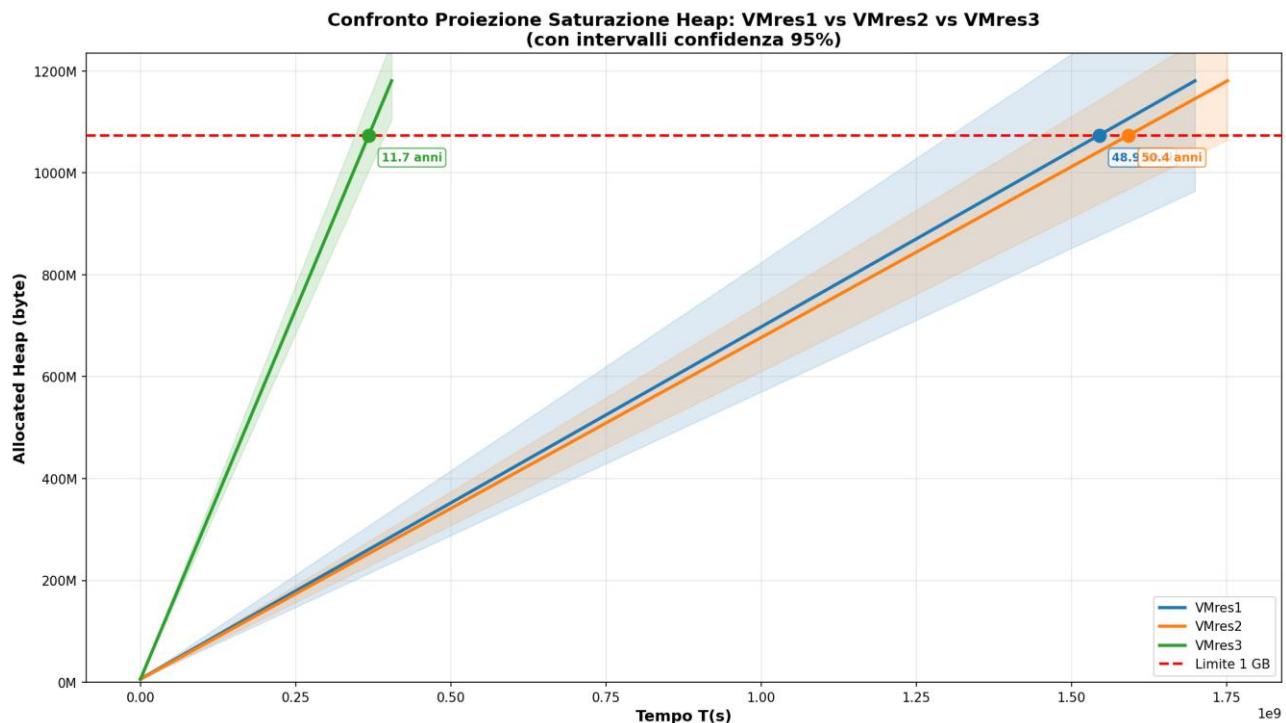
Sheet	Slope	Interval_Low	Interval_Up	Intercept	Trend
VMres1	0.691122	0.563777	0.817824	6562275.46	Crescente
VMres2	0.670950	0.604864	0.736654	5780180.56	Crescente
VMres3	2.903093	2.714760	3.092353	6041585.56	Crescente

Una volta calcolati i parametri regressivi è stato possibile effettuare la failure prediction con tutti e tre i regressori.

Il tempo di fallimento può essere ottenuto come $T_f(s) = \frac{\text{Heap allocato} - \text{intercetta}}{(\text{slope} \pm \text{intervallo})}$, dato che $\text{Heap allocato} = \text{intercetta} + (\text{slope} \pm \text{intervallo}) \cdot T_f(s)$:

- VMres1 $T_f(s) = 48.93 \pm 9.32$
- VMres2 $T_f(s) = 50.44 \pm 5.00$
- VMres3 $T_f(s) = 11.65 \pm 0.76$

Dalla quale si può concludere che **VMres3 è 4.2× più veloce di VMres1/VMres2 nel consumare l'heap**, e la differenza tra il più veloce (VMres3) e il più lento (VMres2) è di 38.8 anni.



5. Design of Experiments

Introduzione

Il “Design of Experiments” è una metodologia sistematica per pianificare test in cui si modificano le variabili di input di un sistema al fine di osservare e identificare le cause dei cambiamenti nell’output.

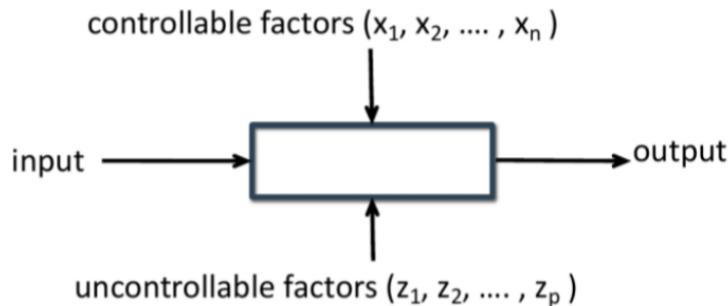


Figura 1. Modello astratto di un tipico sistema a cui viene sottoposta la metodologia DOE

Tra gli obiettivi del DOE ci sono:

- Studiare l’influenza dei fattori controllabili (input) sulla/e uscita/e
- Determinare il/i fattore/i più influente/i
- Studiare le interazioni tra i fattori e la loro influenza sulla/e uscita/e

Questi sono quelli rilevanti per questa sezione del nostro elaborato.

Per una questione di completezza, riportiamo di seguito il formalismo utilizzato durante la trattazione:

- Variabili di risposta: le uscite sulle quali vogliamo controllare l’influenza dei fattori
- Fattori: variabili di input, da scegliere con cura sulla base di ciò che vogliamo valutare
- Livelli: valori che possono assumere i fattori
- Repliche: numero di ripetizione per esperimento

Configurazione

Abbiamo considerato il Web Server Apache HTTP Server 2 come sistema e abbiamo scelto di progettare un **two-factor experiment** con **4 repliche**, dove i fattori sono CTT e Page Type, rispettivamente:

- **CTT**: 25%, 50% e 75% della Usable Capacity (**tre livelli**)
- **Page Type**: dimensione piccola (18 KB), media (0.65 MB) e grande (3 MB), in tutto **tre livelli**

Le **variabili di risposta** scelte sono:

- **Response Time**
- **Latency**
- **Throughput**

Il **numero di esperimenti** totali (osservazioni) ammonta a $3 * 3 * 4 = 36$ per ogni variabile di risposta. Questa sezione dell’elaborato mira ad esporre i risultati dello studio svolto, durante

il quale abbiamo valutato come le prestazioni del sistema (tempo di risposta, latenza e throughput) reagiscono al variare dei parametri di input sopra citati (CTT e dimensione dei file).

Per generare il carico di lavoro abbiamo utilizzato **JMeter**, impostando sessioni da 120 secondi con 50 utenti concorrenti e ramp-up period, ovvero il tempo che JMeter impiega per avviare tutti i 50 thread, di 25 secondi. A valle dei test, abbiamo trasferito i risultati su JMP, costruendo un piano DOE ad hoc per analizzare le metriche ottenute.

	CTT	Page Type	Response Time ...	Throughput [req/s]	Latency [ms]
1	25% (800)	Light	2,7	13,75	1,93
2	25% (800)	Light	2,54	13,74	1,64
3	25% (800)	Light	2,65	13,74	1,73
4	25% (800)	Light	2,5	13,74	1,61
5	25% (800)	Medium	13,82	13,74	1,61
6	25% (800)	Medium	14,11	13,74	1,68
7	25% (800)	Medium	14,28	13,74	1,68
8	25% (800)	Medium	13,5	13,74	1,79
9	25% (800)	Heavy	28,5	13,74	2,21
10	25% (800)	Heavy	28,88	13,74	2,35
11	25% (800)	Heavy	28,56	13,74	2,21
12	25% (800)	Heavy	27,1	13,75	1,99
13	50% (1600)	Light	2,38	27,08	1,59
14	50% (1600)	Light	2,18	27,08	1,36
15	50% (1600)	Light	2,1	27,08	1,28
16	50% (1600)	Light	2,2	27,08	1,37
17	50% (1600)	Medium	11,73	27,08	1,55
18	50% (1600)	Medium	11,58	27,08	1,5
19	50% (1600)	Medium	11,85	27,08	1,48
20	50% (1600)	Medium	13	27,08	1,63
21	50% (1600)	Heavy	31,2	27,05	2,86
22	50% (1600)	Heavy	30,25	27,03	2,71
23	50% (1600)	Heavy	21,77	27,05	1,94
24	50% (1600)	Heavy	22,85	27,07	1,85
25	75% (2400)	Light	2,63	40,41	1,77
26	75% (2400)	Light	2,69	40,41	1,85
27	75% (2400)	Light	2,38	40,41	1,62
28	75% (2400)	Light	2,41	40,41	1,53
29	75% (2400)	Medium	12,43	40,4	1,55
30	75% (2400)	Medium	12,36	40,4	1,75
31	75% (2400)	Medium	14,42	40,4	2,08
32	75% (2400)	Medium	13,08	40,41	2,29
33	75% (2400)	Heavy	22,08	40,39	1,66
34	75% (2400)	Heavy	20,06	40,4	1,41
35	75% (2400)	Heavy	19,23	40,39	1,34
36	75% (2400)	Heavy	19,61	40,39	1,32

Figura 2. Tabella JMP riportante i risultati degli esperimenti

Analisi di Importanza

Prima di riportare i risultati e le tecniche adottate per il calcolo dell'importanza dei fattori, è importante osservare che **l'importanza non è un fattore statistico ma un parametro soggettivo**. Infatti, siamo noi a scegliere se un fattore è importante o meno a seconda dei risultati ottenuti in relazione alla Sum of Squares Total (SST).

La tecnica adottata per valutare l'importanza dei fattori è la stima del modello sul piano DOE. Abbiamo effettuato una regressione lineare con l'obiettivo di minimizzare l'MSE ed enfasi sul leverage degli effetti, ovvero su quanto un singolo fattore (o interazione) è capace di influenzare le variabili di uscita, ottendo i risultati riportati nelle figure:

Analisi della varianza					
Origine	DF	Somma dei quadrati	Media quadratica	Rapporto F	Prob > F
Modello	8	3207,6619	400,958	130,3485	
Errore	27	83,0532	3,076	Prob > F	<.0001*
C. totale	35	3290,7151			

Stime dei parametri					
Test degli effetti					
Origine	Nparm	DF	Somma dei quadrati	Rapporto F	Prob > F
CTT		2	53,4684	8,6911	0,0012*
Page Type		2	3058,0227	497,0708	<.0001*
CTT*Page Type		4	96,1707	7,8161	0,0003*

Figura 3.a. Analisi degli effetti per Y=Response Time

Analisi della varianza					
Origine	DF	Somma dei quadrati	Media quadratica	Rapporto F	Prob > F
Modello	8	4264,5367	533,067	13084374	
Errore	27	0,0011	4,074e-5	Prob > F	<.0001*
C. totale	35	4264,5378			

Stime dei parametri					
Test degli effetti					
Origine	Nparm	DF	Somma dei quadrati	Rapporto F	Prob > F
CTT		2	4264,5336	52337458	<.0001*
Page Type		2	0,0017	20,5227	<.0001*
CTT*Page Type		4	0,0014	8,3523	0,0002*

Figura 3.b. Analisi degli effetti per Y=Throughput

Analisi della varianza					
Origine	DF	Somma dei quadrati	Media quadratica	Rapporto F	Prob > F
Modello	8	3,3639000	0,420488	7,6530	
Errore	27	1,4835000	0,054944	Prob > F	<.0001*
C. totale	35	4,8474000			

Stime dei parametri					
Test degli effetti					
Origine	Nparm	DF	Somma dei quadrati	Rapporto F	Prob > F
CTT		2	0,2146167	1,9530	0,1614
Page Type		2	0,9230167	8,3995	0,0015*
CTT*Page Type		4	2,2262667	10,1296	<.0001*

Figura 3.c. Analisi degli effetti per Y=Latency

L'analisi si è concentrata sui dati derivanti dall'**analisi della varianza** (ANOVA) e dal **test degli effetti**. Questi strumenti sono stati fondamentali per quantificare il peso (*importanza*) dei singoli fattori e degli errori su ciascuna variabile di risposta. Il metodo utilizzato per determinare l'importanza si basa sulla **Sum of Squares** (SS), e in un two-factor experiment la varianza totale viene esplicitata come segue:

$$SST = SSA + SSB + SSAB + SSE$$

Dove:

- **SST**: Varianza totale
- **SSA & SSB**: Varianze dovute ai singoli fattori
- **SSAB**: Varianza dovuta all'interazione tra i due fattori
- **SSE**: Sum of Squares Error (Errore residuo)

L'importanza si calcola normalizzando la singola Sum of Square rispetto a quella totale:

$$Imp(B) = \frac{SSB}{SST}$$

Riportiamo di seguito in una tabella il risultato dell'Analisi di Importanza per ogni variabile di uscita.

Fattore\Var. di risposta	Response Time	Latency	Throughput
CTT	$\frac{SSCTT}{SST} = \frac{53.4684}{3290.7151} = 1.6\%$	$\frac{SSCTT}{SST} = \frac{0.2146}{4.8474} = 4.4\%$	$\frac{SSCTT}{SST} = \frac{4264.5367}{4264.5378} = 99.99\%$
Page Type	$\frac{SSPT}{SST} = \frac{3058.0227}{3290.7151} = 92.9\%$	$\frac{SSPT}{SST} = \frac{0.9230}{4.8474} = 19.1\%$	< 0.01%
CTT * Page Type	$\frac{SSINT}{SST} = \frac{96.1707}{3290.7151} = 3.0\%$	$\frac{SSINT}{SST} = \frac{2.2263}{4.8474} = 45.9\%$	< 0.01%
Errore	$\frac{SSE}{SST} = \frac{83.0532}{3290.7151} = 2.5\%$	$\frac{SSE}{SST} = \frac{1.4835}{4.8474} = 30.6\%$	< 0.01%

Tabella 1. Risultato dell'Analisi di importanza per ogni fattore e per l'errore, per le singole uscite.

P.S: Qui evince un "difetto" nella selezione del tipo di pagine per l'analisi del Throughput: il CTT è troppo piccolo, ergo il carico non impatta sufficientemente sul Throughput). Procederemo comunque con l'analisi.

I risultati ottenuti sono i seguenti:

1. Per il **Response Time**, è il **Page Type** ad avere la maggiore importanza.
2. Per la **Latency**, invece, l'**interazione** tra i fattori è sicuramente la più importante, senza però poter trascurare l'impatto dell'**errore**.
3. Per il **Throughput**, invece, il **CTT** è nettamente il più importante tra tutti.

Analisi di Significatività

Per procedere a tale analisi, è stato necessario utilizzare dei test statistici, in quanto la **significatività** è un **fattore statistico**! Essa è il **contributo alla variazione rispetto all'errore**. Tuttavia, è necessario **determinare normalità dei residui delle variabili di risposta** (errori sulla stima del modello dovuti alle replicate degli esperimenti) e **omoschedasticità** di questi **rispetto ai fattori**, che sono le assunzioni fondamentali da verificare per individuare il corretto metodo ANOVA da applicare.

Di seguito sono riportate le colonne dei residui delle variabili di risposta:

Residuo Response Time (elapsed) [ms]	Residuo Throughput [req/s]	Residuo Latency [ms]
0,1025	0,0075	0,2025
-0,0575	-0,0025	-0,0875
0,0525	-0,0025	0,0025
-0,0975	-0,0025	-0,1175
-0,1075	1,24345e-14	-0,08
0,1825	1,24345e-14	-0,01
0,3525	1,24345e-14	-0,01
-0,4275	1,24345e-14	0,1
0,24	-0,0025	0,02
0,62	-0,0025	0,16
0,3	-0,0025	0,02
-1,16	0,0075	-0,2
0,165	-7,10543e-15	0,19
-0,035	-7,10543e-15	-0,04
-0,115	-7,10543e-15	-0,12
-0,015	-7,10543e-15	-0,03
-0,31	-3,55271e-15	0,01
-0,46	-3,55271e-15	-0,04
-0,19	-3,55271e-15	-0,06
0,96	-3,55271e-15	0,09
4,6825	-3,55271e-15	0,52
3,7325	-0,02	0,37
-4,7475	-3,55271e-15	-0,4
-3,6675	0,02	-0,49
0,1025	0	0,0775
0,1625	0	0,1575
-0,1475	0	-0,0725
-0,1175	0	-0,1625
-0,6425	-0,0025	0,3675
-0,7125	-0,0025	-0,1675
1,3475	-0,0025	0,1625
0,0075	0,0075	0,3725
1,835	-0,0025	0,2275
-0,185	0,0075	-0,0225
-1,015	-0,0025	-0,0925
-0,635	-0,0025	-0,1125

Figura 4. Sezione della tabella sul sw JMP relativa ai residui delle variabili di risposta.

Analisi Normalità

Per procedere all'analisi della normalità, abbiamo visualizzato le distribuzioni dei residui per effettuare un primo test visivo tramite plot quantile-quantile.

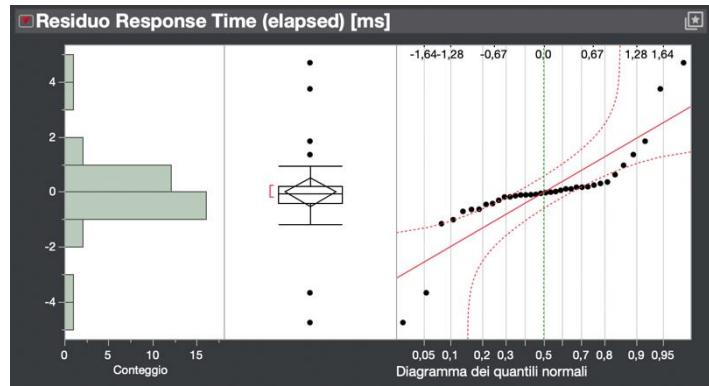


Figura 5.a. Distribuzione del residuo di Response Time

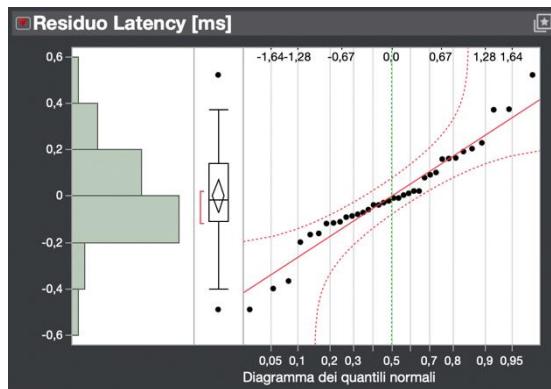


Figura 5.b. Distribuzione del residuo di Latency

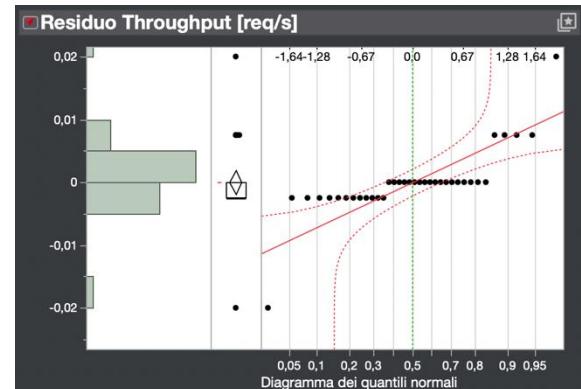


Figura 5.c. Distribuzione del residuo di Throughput

Dall'analisi visuale si evince che:

- Response Time:** La **normalità** è rispettata nella **parte centrale** del plot quantile, ma si notano **forti deviazioni** sulle **code** che escono dalle bande di confidenza
- Latency:** È la distribuzione che meglio approssima la **normalità**, essendo i residui ben allineati sulla retta di riferimento e contenuti quasi interamente nelle bande di confidenza
- Throughput:** Deviazioni marcate, specialmente agli estremi, indicano una evidente **mancanza di normalità**.

Per chiarire il dubbio sul Response Time, è possibile applicare il **Test di Shapiro-Wilk**.

Di seguito sono riportati i risultati del Test per Response Time e Latency (inutile per Throughput).

▼ Test della bontà di adattamento			▼ Test della bontà di adattamento			
	W	Prob<W		W	Prob<W	
Shapiro-Wilk	0,789006	<,0001*		Shapiro-Wilk	0,967212	0,3544
	A ²	P-value simulato			A ²	P-value simulato
Anderson-Darling	3,285416	<,0001*	Anderson-Darling	0,5518546	0,1412	

Figura 6. Test Shapiro-Wilk rispettivamente su residuo Response Time e su residuo Latency

Sia il test visivo sia quello di Shapiro-Wilk ci dicono che per il **residuo** di **Response Time non ho la normalità**, mentre per la **Latency sì!**

Analisi Omoschedasticità

Prima di procedere all'analisi, è importante osservare che da questa possiamo **escludere le variabili di uscita** sui quali **residui** abbiamo determinato la **non normalità**, ovvero Response Time e Throughput, poiché per queste basta applicare il Test di Wilcoxon/Kruskal-Wallis alle stesse variabili di risposta per ogni fattore.

L'unica variabile di risposta oggetto di questa analisi, dunque, è la **Latency**.

Riportiamo di seguito i risultati dei test di verifica dell'omoschedasticità per la Latency rispetto ai fattori.

Test	Rapporto F	Num DF	Den DF	Prob > F	Test	Rapporto F	Num DF	Den DF	Prob > F
O'Brien[.5]	2,3591	2	33	0,1103	O'Brien[.5]	2,9651	2	33	0,0654
Brown-Forsythe	1,7926	2	33	0,1824	Brown-Forsythe	2,5832	2	33	0,0907
Levene	2,2251	2	33	0,1240	Levene	2,5464	2	33	0,0937
Bartlett	3,7926	2	.	0,0225*	Bartlett	3,6701	2	.	0,0255*

Figura 7. Test per verificare l'omoschedasticità di Latency rispetto a CTT e Page Type

Dai test, risulta evidente che abbiamo omoschedasticità per Latency per entrambi i fattori, ergo è necessario applicare l'F-Test per studiare la significatività dei fattori (Figura 3.c).

Per quanto riguarda Response Time e Throughput, di seguito sono riportati i risultati del Test di Wilcoxon/Kruskal-Wallis per ogni fattore.

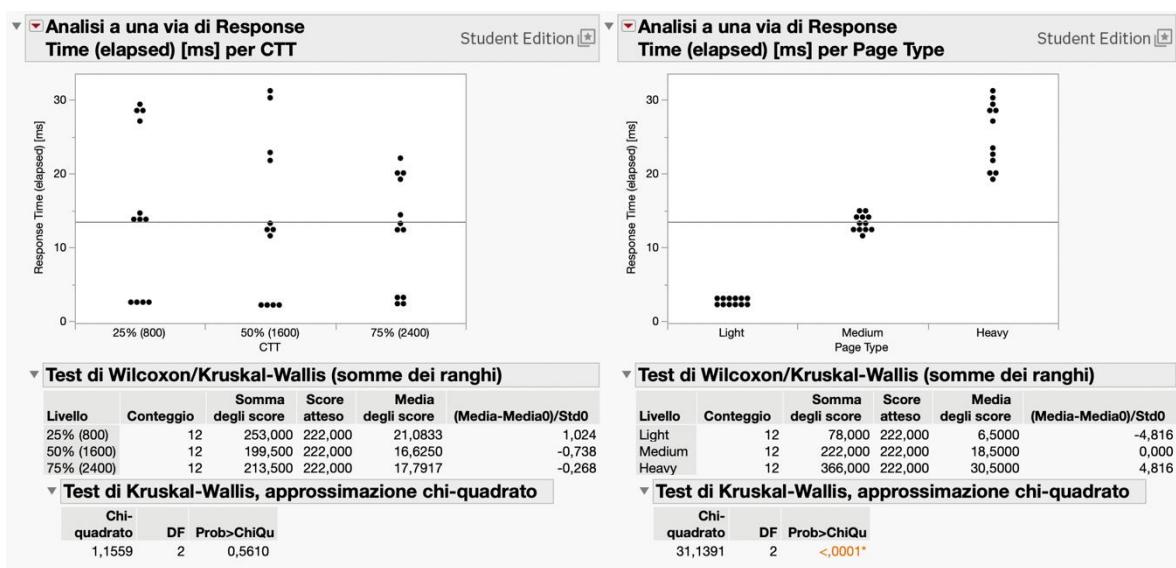


Figura 8.a. Test di Wilcoxon/Kruskal-Wallis per Response Time, rispettivamente con CTT e Page Type

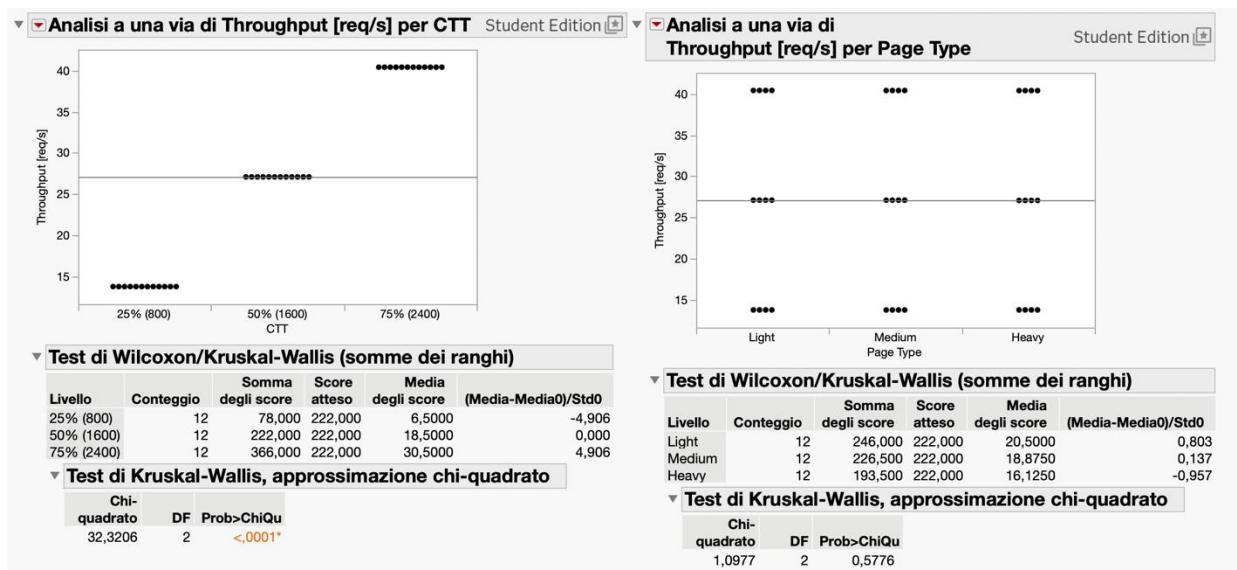


Figura 8.b. Test di Wilcoxon/Kruskal-Wallis per Throughput, rispettivamente con CTT e Page

Conclusioni

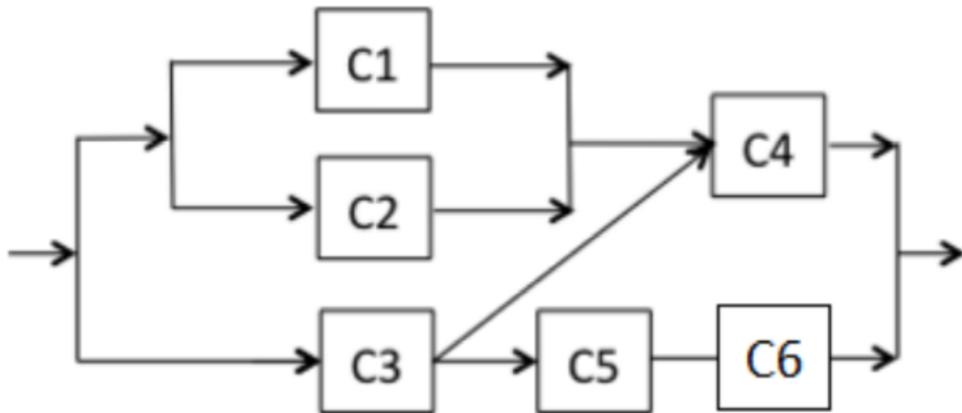
Sulla base dei risultati ottenuti, possiamo concludere che:

- **Response Time:** il fattore più importante risulta essere Page Type, il quale risulta essere anche significativo, mentre CTT non risulta essere né significativo né importante.
- **Latency:** l'interazione tra i due fattori risulta essere sia importante sia significativa, mentre l'errore risulta essere importante e il Page Type significativo.
- **Throughput:** Il fattore CTT è indiscutibilmente il più importante e significativo.

6. Esercizi Dependability

Esercizio 1

Find the $R(t)$ and MTTF for the system whose reliability diagram is given below. In calculating MTTF, assume all components are identical and fail randomly with failure rate λ .



Svolgimento:

In questo esercizio è richiesto di calcolare la reliability del sistema complessivo $R_{sys}(t)$ (per semplicità, indicata semplicemente come R_{sys}) e il MTTF.

A partire dal diagramma a blocchi, è evidente la presenza del parallelo tra C1 e C2, il che consente di calcolare facilmente la reliability associata a quella porzione, così come la serie tra C5 e C6.

Per il parallelo tra C1 e C2, è sufficiente applicare la formula della configurazione parallela:

$$R_{1,2} = 1 - \prod_{i=1}^2 (1 - R_i) = 1 - (1 - R_1)(1 - R_2)$$

Mentre dalla serie tra C5 e C6 risulta:

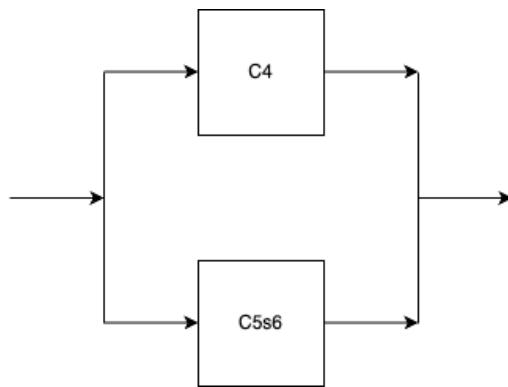
$$R_{5,6} = \prod_{i=5}^6 R_i = R_5 R_6$$

Tuttavia, il collegamento tra l'output di C3 e l'input di C4 ci costringe a definire l'architettura come una **NON Serial-Parallel RBD!** Per risolvere questo esercizio, è sufficiente applicare la tecnica del **Conditioning** ad uno dei due nodi tra C3 e C4: scegliamo per semplicità di disegno C3.

P.S: Ricordiamo la formula del conditioning, scelto m come il sistema *conditioned*:

$$R_{sys} = P\{\text{system is working} \mid m \text{ is UP}\} * R_m + P\{\text{system is working} \mid m \text{ is DOWN}\} * (1 - R_m)$$

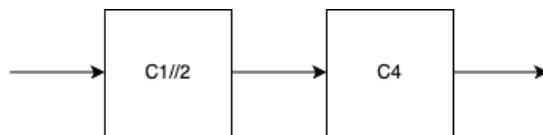
Scegliamo di partire dal sistema dove C3 è UP (funziona):



La Reliability di questo circuito è facilmente calcolabile come:

$$R_{4,5,6} = 1 - (1 - R_4)(1 - R_{5,6}) = 1 - (1 - R_4)(1 - R_5 R_6)$$

Segue il caso in cui C3 viene considerato DOWN (non funzionante) e il calcolo della Reliability:



$$R_{1,2,4} = R_{1,2} R_4 = [1 - (1 - R_1)(1 - R_2)] R_4$$

Siccome dalla traccia sappiamo che tutte le componenti hanno la stessa reliability, la indichiamo semplicemente R . Usando la formula del Conditioning, otteniamo la R_{sys} :

$$\begin{aligned} R_{sys} &= R_{4,5,6} * R + R_{1,2,4} * (1 - R) = \\ &= [1 - (1 - R)(1 - R^2)] * R + [(1 - (1 - R)^2)R] * (1 - R) = \\ &= -2R^3 + 3R^2 = -2e^{-3\lambda t} + 3e^{-2\lambda t} \end{aligned}$$

Dove la forma esponenziale deriva dal fatto che R ha un failure rate di tipo esponenziale ($\sim Exp(\lambda)$).

A questo punto abbiamo calcolato il MTTF a partire dalla formula:

$$MTTF = \int_0^{+\infty} t f(t) dt = - \int_0^{+\infty} t R'(t) dt = \int_0^{+\infty} R(t) dt = \int_0^{+\infty} (-2e^{-3\lambda t} + 3e^{-2\lambda t}) dt = \frac{5}{6\lambda}$$

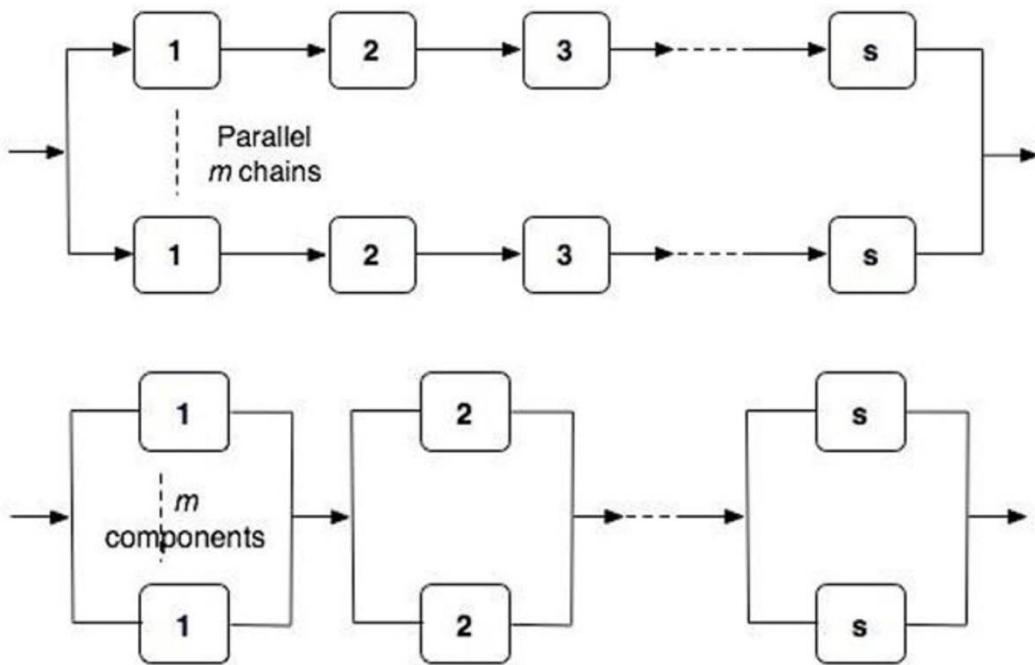
Questo risultato evidenzia come il **MTTF del sistema sia inferiore a quello di un singolo componente**, risultando quindi **meno affidabile (reliable)**!

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).



Svolgimento:

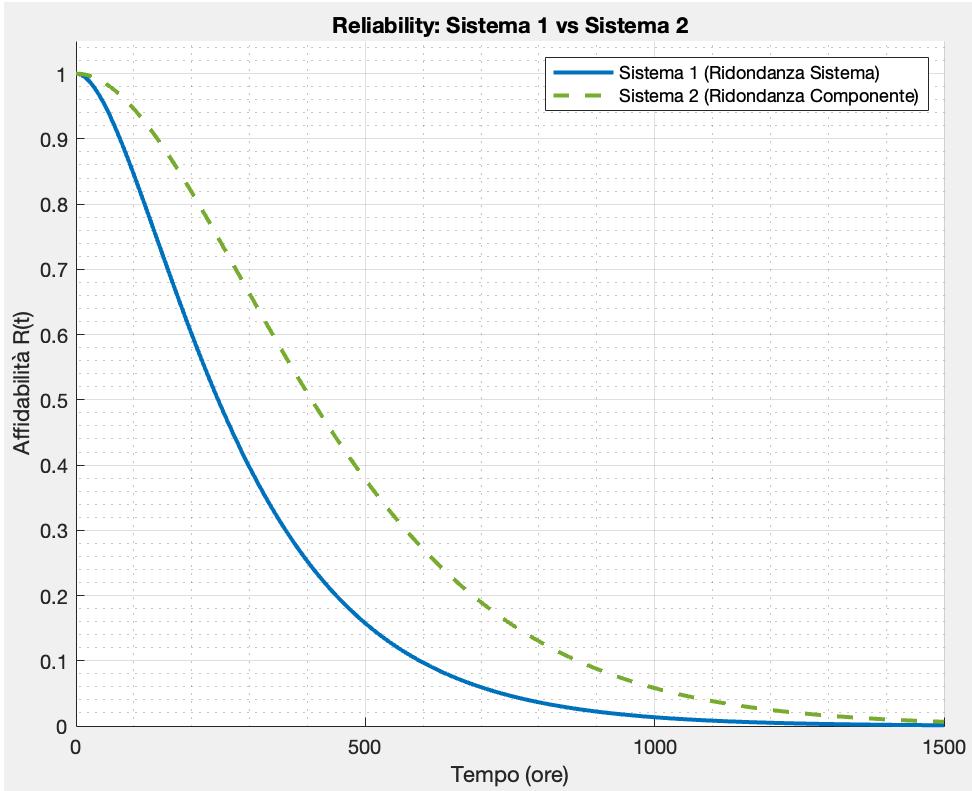
La reliability dei due sistemi può essere calcolata come segue:

$$R_{S_1} = 1 - \prod_{i=1}^m \left(1 - \prod_{j=1}^s R_j \right) = 1 - (1 - R^s)^m$$

$$R_{S_2} = \prod_{i=1}^s \left(1 - \prod_{k=1}^m (1 - R_k) \right) = [1 - (1 - R)^m]^s$$

Da una prima analisi visiva, è evidente che $R_{S_2} > R_{S_1}$, poiché la ridondanza a livello di componente (Sist. 2) è sempre più affidabile della ridondanza a livello di sistema (Sist. 1), in quanto permette di aggirare un guasto singolo senza dover "sacrificare" l'intero success path.

Di seguito è riportato il grafico delle due R_S , ricordando che $MTTF = 800h$, $m = 2$, $s = 4$ e andando a scrivere $R_i = e^{-\lambda t} = e^{-\frac{t}{800}}$:



Per rispondere al quesito posto, è necessario capire quanti percorsi sequenziali replicati sono necessari per il Sistema 1, a parità di componenti, per raggiungere la reliability del Sistema 2. Per fare ciò, uguagliamo le due formule, lasciando come unica incognita m_1 per il Sistema 1:

$$1 - (1 - R^4)^{m_1} = [1 - (1 - R)^2]^4$$

Da cui otteniamo:

$$m_1 = \frac{\ln[1 - (1 - (1 - R)^2)^4]}{\ln(1 - R^4)}$$

A questo punto, per poter calcolare m_1 ed R selezioniamo l'istante $t = MTTF = 800h$, e otteniamo (mediante Matlab):

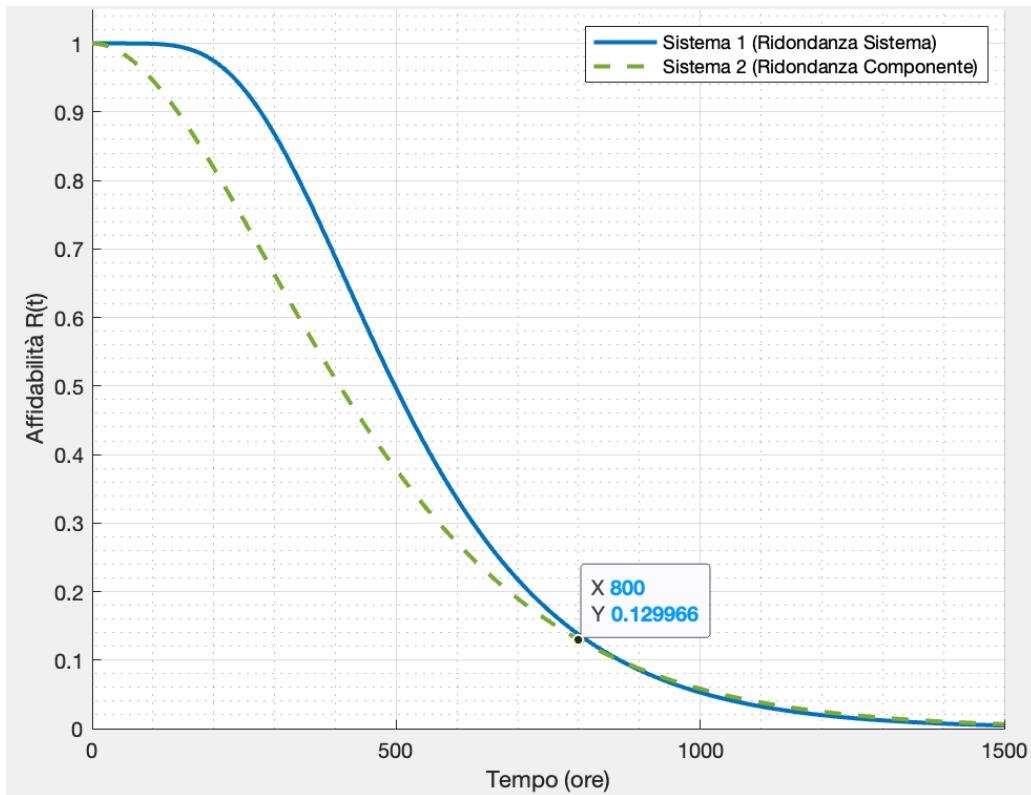
$$R \approx 0.368$$

$$m_1 = \frac{\ln[1 - (1 - (1 - 0.368)^2)^4]}{\ln(1 - (0.368)^4)} = 7.55 \approx 8$$

Da notare come se avessimo scelto una R più piccola, ovvero un t più grande, avremmo ottenuto una m_1 più grande, segno che un'affidabilità minore comporta una maggiore ridondanza a livello di sistema per compensare.

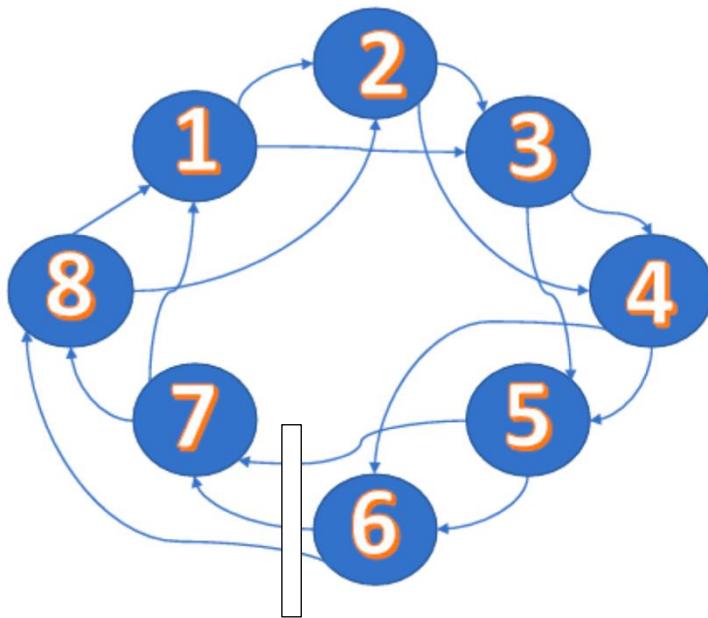
Precisiamo, inoltre, che la m del **Sistema 2 rimane invariata** ($m_2 = m = 2$)!

Infine, abbiamo ricalcolato R_{S_1} e R_{S_2} rispettivamente con $m_1 = 8$ e $m_2 = 2$, e abbiamo verificato che il nostro operato ha portato al risultato auspicato (*nel punto $t = 800h$ le curve coincidono*):



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.



Svolgimento:

Affinché questa rete funzioni, devono essere attivi almeno 4 nodi su 8 senza che si verifichino guasti consecutivi (si guastano due nodi consecutivi). Assumendo che tutti i componenti abbiano la stessa affidabilità R_m (omettendo anche qui per semplicità la dipendenza dal tempo), possiamo trattare il sistema come un 'M-out-of-N' la cui **Reliability** sarà:

$$R_{sys} = \sum_{i=0}^{N-M} \binom{N}{i} R_m^{N-i} (1-R_m)^i = \sum_{i=0}^4 \binom{8}{i} R_m^{8-i} (1-R_m)^i$$

Precisiamo che le combinazioni che non portano ad un sistema funzionante dovranno essere escluse! Di seguito è riportata l'analisi dei possibili scenari di guasto, dove con K sono riportati i nodi fuori uso, per isolare le configurazioni con nodi adiacenti guasti:

- **$K = 0, 1$** : Il sistema è sicuro => **Combinazioni non funzionanti = 0**;
- **$K = 2$** : Su $\binom{8}{2} = 28$ combinazioni totali, le coppie adiacenti possibili sono esattamente **8** (1-2, 2-3, 3-4, 4-5, 6-7, 7-8, 8-1), le quali portano al fallimento del sistema;

- **K = 3:** Su $\binom{8}{3} = 56$ combinazioni totali, escludendo quelle valide, si ottengono **40** configurazioni non funzionanti (presentano almeno una coppia adiacente guasta); questo numero è ottenuto osservando che il sistema fallisce se falliscono 3 nodi consecutivi (8 combinazioni) o se falliscono 2 nodi consecutivi più un nodo spaiato (4 combinazioni per il nodo spaiato per ogni coppia di nodi consecutivi, le coppie possibili di nodi consecutivi sono 8 quindi $8 \times 4 = 32$);
- **K = 4:** Su $\binom{8}{4} = 70$ combinazioni totali, solo 2 garantiscono il funzionamento (alternanza perfetta pari/dispari, non ho nodi consecutivi guasti). Di conseguenza, sono ben **68** le configurazioni che portano al fallimento;
- **K ≥ 5:** Il sistema fallisce sempre poiché non viene raggiunto il numero minimo di nodi attivi ($M = 4$). Le combinazioni totali in questi casi sono, rispettivamente, **56, 28, 8 e 0**.

La **Unreliability** totale del sistema F_{sys} è stata calcolata utilizzando la Legge della Probabilità Totale, dove $i \geq 2$ perché per $i = 0, 1$ il sistema è sicuro:

$$F_{sys} = \sum_{i=2}^{8} P(\text{System Failed} | i \text{ nodes failed}) * P(i \text{ nodes failed}) = \\ = 8R_m^6(1 - R_m)^2 + 40R_m^5(1 - R_m)^3 + 68R_m^4(1 - R_m)^4 + 56R_m^3(1 - R_m)^5 + 28R_m^2(1 - R_m)^6 \\ + 8R_m(1 - R_m)^2 + (1 - R_m)^8$$

Sapendo che $R_{sys} = 1 - F_{sys}$, assumendo come traccia $R_m(t) = e^{-\lambda t}$ e $\lambda = 0.005/h$, abbiamo infine calcolato la **Reliability** del sistema **dopo 48 ore**:

$$R_{sys}(48h) \approx 0.729$$

Esercizio 4

A data center relies on a cooling system to maintain optimal operating temperatures for its servers. The cooling system is composed of the following components:

- Main Cooling Unit (C): Provides primary cooling. MTTF_C=10000h
- Backup Cooling Unit (B): Activates only if the Main Cooling Unit fails. MTTF_B=5000h
- Power Supply (P): Provides power to the cooling units. MTTF_P=7000h
- Control Panel (CP): Monitors and manages the cooling process. MTTF_{CP}=6000h

The cooling system operates as follows.

The Main Cooling Unit and the Power Supply must work together in series for primary cooling. If the Main Cooling Unit fails, the Backup Cooling Unit activates, provided the Power Supply is still operational.

The Control Panel must always function.

1) Draw the RBD.

Represent the system as an RBD, showing the series and parallel arrangements, including the redundancy of the cooling units.

2) Calculate System Reliability.

Compute the overall reliability of the cooling system.

3) Impact of Redundancy:

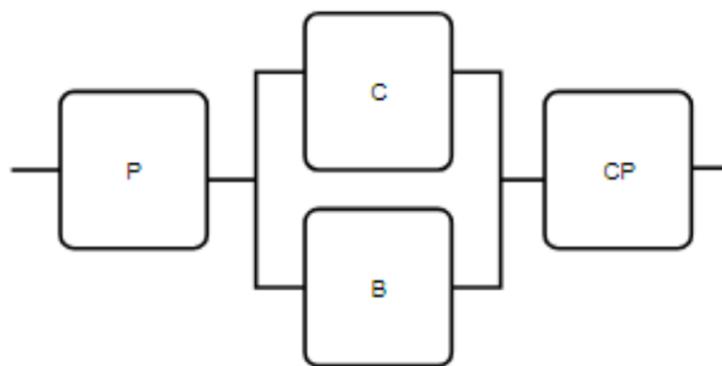
Discuss how the addition of the Backup Cooling Unit affects the system reliability compared to a simplex system (without redundancy for the cooling units).

Calculate the system reliability and visualize how the reliability changes with varying MTTF_B values.

Svolgimento:

Punto 1:

Di seguito è riportato il **RBD** del sistema in esame:



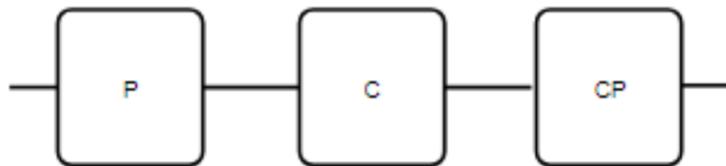
Punto 2:

L'espressione analitica della **Reliability** del sistema di raffreddamento è data da:

$$R_{sys} = R_P \left(1 - (1 - R_C)(1 - R_B)\right) R_{CP}$$

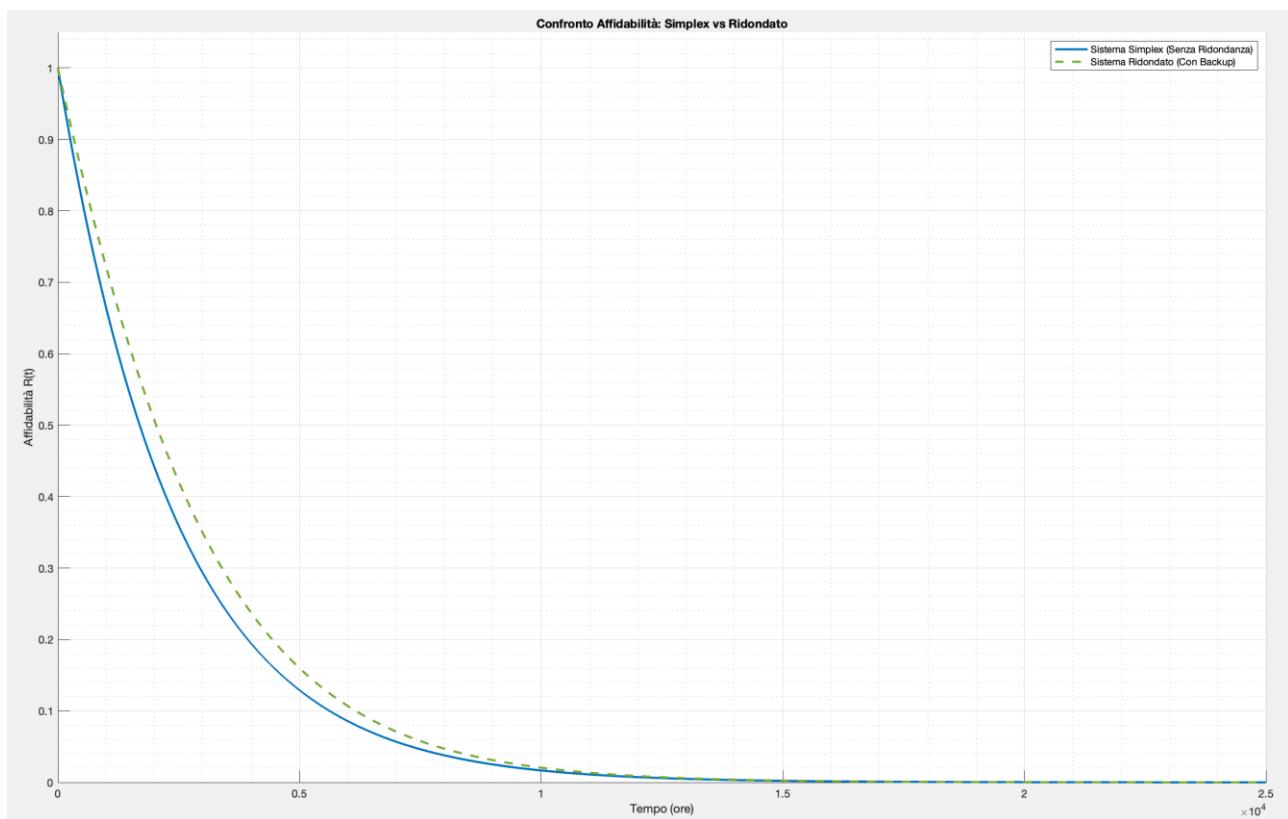
Punto 3:

Indichiamo come “**simplex**” il **sistema** che **non contempla** la presenza del **Backup Cooling (B)**. Di seguito sono riportati il **RBD** e la Reliability R_{simplex} del sistema **simplex**:



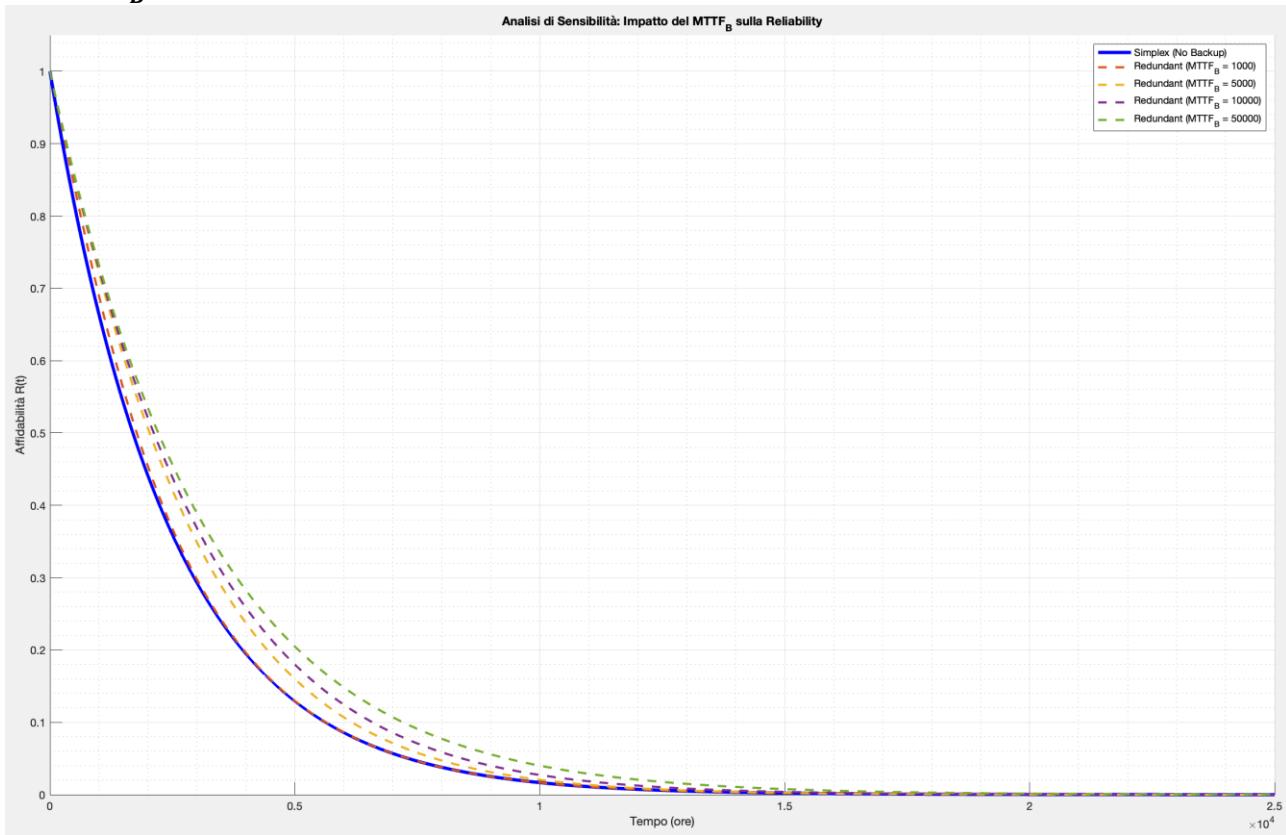
$$R_{\text{simplex}} = R_P R_C R_{CP}$$

Conoscendo i $MTTF$ e sapendo che le reliability di questi componenti hanno un failure rate esponenziale ($\sim \text{Exp}(\frac{1}{MTTF_i})$), abbiamo realizzato uno script Matlab per graficare l’andamento di R_{sys} e di R_{simplex} (ricordando che $R_i(t) = e^{-\frac{t}{MTTF_i}}$), ottenendo il seguente risultato:



È evidente che, mentre all’inizio esiste una differenza sostanziale a favore della configurazione ridondata, questa si assottiglia fino a scomparire dopo $1.5 * 10^4$ ore di operatività. Il motivo è che l’affidabilità globale rimane vincolata ai componenti in serie: avendo questi un $MTTF$ non elevato, finiscono per compromettere la tenuta del sistema. L’approccio ideale prevederebbe quindi di ridondare anche questi ultimi elementi critici.

Vediamo adesso, dall'analisi da noi effettuata, il **comportamento** di R_{sys} e $R_{simplex}$ al variare di $MTTF_B$:

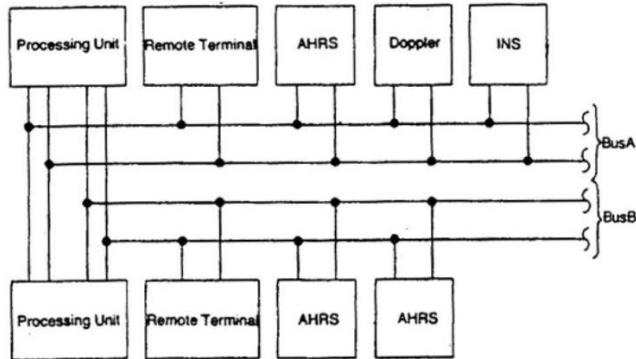


Dall'analisi di sensibilità è emerso che il miglioramento del $MTTF_B$ sposta la curva di affidabilità verso l'alto, ma con un effetto di **saturazione**. Individuiamo di seguito i limiti:

- **Limite Inferiore:** Se il $MTTF_B$ fosse molto basso ($\leq 1000h$), il backup si guasterebbe quasi subito, rendendo il sistema ridondato quasi identico al Simplex (le curve si sovrappongono velocemente).
- **Limite Superiore:** Anche ipotizzando un backup perfetto ($MTTF_B \rightarrow \infty$) l'affidabilità del sistema non tende a 1, ma è asintoticamente limitata dalla affidabilità dei componenti in serie. Questo conferma che, oltre una certa soglia di qualità del backup, non ha senso investire ulteriormente su di esso senza migliorare l'affidabilità di Power Supply e Control Panel ($MTTF_P$ e $MTTF_{CP}$)!

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.

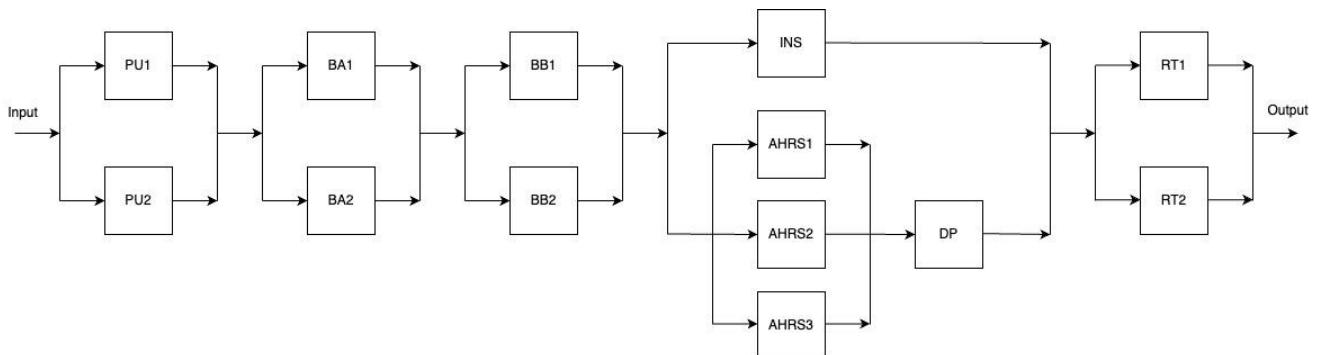
Equipment	MTTF (hr)
Processing Unit	10000
Remote Terminal	4500
AHRS	2000
INS	2000
Doppler	500
Bus	60000

- 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.

Svolgimento:

Punto a:

Di seguito è riportato il **Reliability Block Diagram** del sistema complessivo:

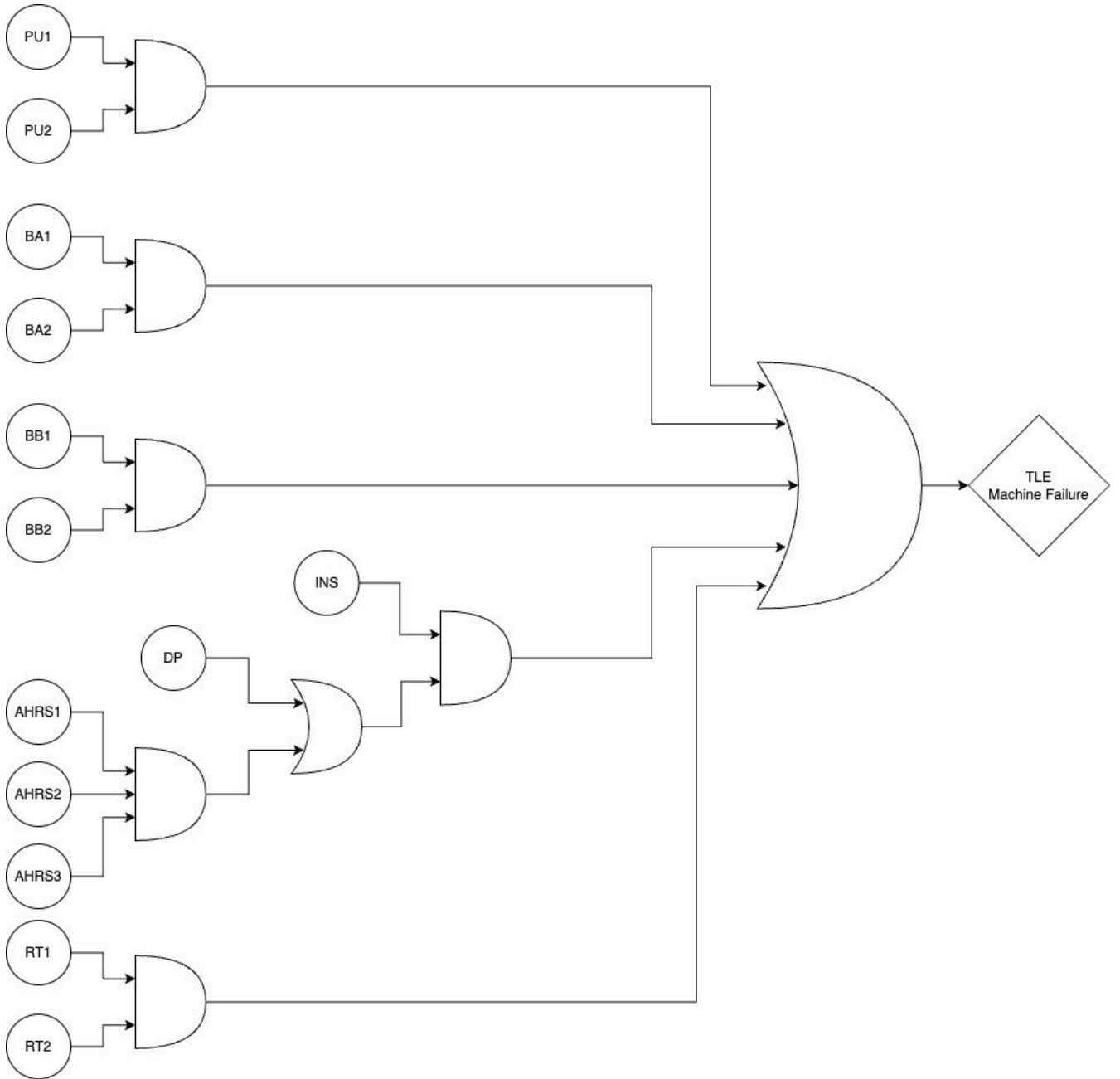


Punto b:

Per realizzare un Fault Tree per il sistema in analisi, abbiamo utilizzato le tecniche di conversione da RBD a Fault Tree:

- **Parallelo → porta OR**
- **Serie → porta AND**

Di seguito lo schema ottenuto:



Per individuare i minimal cutsets è stato necessario considerare la funzione di uscita:

$$\Phi(x) = (PU_1 \wedge PU_2) \vee (BA_1 \wedge BA_2) \vee (BB_1 \wedge BB_2) \vee (INS \wedge (DP \vee (AHRS_1 \wedge AHRS_2 \wedge AHRS_3))) \vee (RT_1 \wedge RT_2)$$

I **minimal cutsets** sono dunque i seguenti:

- $PU_1 \wedge PU_2$

- $BA_1 \wedge BA_2$
- $BB_1 \wedge BB_2$
- $INS \wedge (DP \vee (AHRS_1 \wedge AHRS_2 \wedge AHRS_3))$
- $RT_1 \wedge RT_2$

Punto c:

Per calcolare la Reliability del sistema dopo un'ora di viaggio è sufficiente calcolare $R_{sys}(t)$, assumendo distribuzione esponenziale dei fallimenti e coverage 100% dei guasti, e poi stabilire $t = 1h$:

$$R_{sys}(t) = (1 - (1 - R_{PU})^2)(1 - (1 - R_{BUS})^2)^2(1 - (1 - R_{INS})(1 - R_{DP}(1 - (1 - R_{AHRS})^3))(1 - (1 - R_T)^2)$$

$$R_{sys}(1h) \approx 0.99999 \quad [5 \text{ nines}]$$

Punto d:

Inserendo un fattore di coverage c incognito nella detezione e correzioni di errori relativi alla processing unit, la $R_{sys}(t)$ diventa:

$$R_{sysc}(t) = (1 - (1 - R_{PU})(1 - R_{PU} * c))(1 - (1 - R_{BUS})^2)^2(1 - (1 - R_{INS}) \\ (1 - R_{DP}(1 - (1 - R_{AHRS})^3))(1 - (1 - R_T)^2)$$

Dove $R_{PUmod}(t) = 1 - (1 - R_{PU})(1 - R_{PU} * c)$ riflette la presenza del fattore di coverage: il successo del sistema è garantito dal funzionamento dell'unità primaria o, in alternativa, dal funzionamento del suo backup qualora ci sia un guasto che coinvolge l'unità primaria e quest'ultimo venga rilevato correttamente!

Infatti, $R_{PUmod}(t)$ si può riscrivere come:

$$R_{PUmod}(t) = 1 - (1 - R_{PU})(1 - R_{PU} * c) = R_{PU} + cR_{PU}(1 - R_{PU})$$

Per calcolare il valore di c è sufficiente imporre $R_{sysc}(1h) = 0.99999$, ottenendo di conseguenza:

$$c \approx 0.91008$$

7. FFDA: analisi file di log

La Field Failure Data Analysis, in sigla FFDA, è un insieme di tecniche di misura diretta il cui obiettivo è andare a misurare direttamente una serie di attributi di dependability dei sistemi. È dunque un particolare tipo di data-driven analysis, cioè un'analisi basata su dati sul campo, relativa a fallimenti accidentali o non accidentali, come ad esempio gli attacchi. In questo ambito risulta particolarmente importante comprendere in che modo i fallimenti si vanno a verificare sul campo, quindi andiamo a monitorare il sistema, cerchiamo di comprendere come esso si comporta e in che modo il sistema manifesta i fallimenti per fare una serie di stime andando ad ottenere TTF empirici e Reliability empiriche. Il vantaggio principale offerto da questa tecnica è che le misure sono dirette. Gli obiettivi principali della FFDA sono quindi identificare le classi di errore analizzando come esse si manifestano sul campo, in modo tale da poter misurare le reliability in sistemi operazionali; inoltre, vogliamo andare ad analizzare le distribuzioni statistiche dei tempi al fallimento e al ripristino e studiare la correlazione che vige tra i fallimenti e il workload applicato al sistema. Infine, si cerca tramite questa analisi di andare a identificare le cause principali delle interruzioni dando una serie di indicazioni riguardo i bottleneck per la dependability e di fornire dati utili per la validazione e la popolazione di failure models simulati.

7.1 Mercury

Mercury è il primo cluster che prendiamo in considerazione per l'analisi FFDA; esso è costituito da nodi IBM ed è realizzato basandosi su un'architettura a tre livelli, più un nodo di gestione definito tg-master. I tre livelli sono login, identificato con la nomenclatura tg-loginX, calcolo, definiti come tg-cX, e infine storage, con notazione tg-sX dove la X indica un valore numerico intero che identifica il nodo all'interno dello specifico livello. Le interconnessioni tra i livelli e i nodi sono gestite da un'architettura Myrinet e ogni nodo ha in esecuzione un sistema con SO RedHat 9.0.

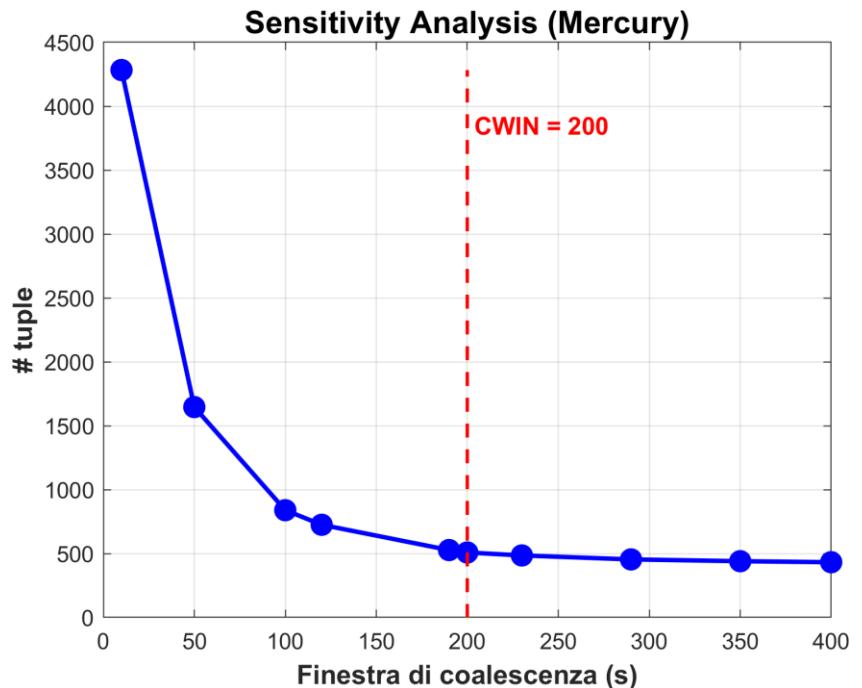
I valori analizzati sono ottenuti tramite il daemon syslogd, ed inoltre il log è stato filtrato per ottenere un file che riporti solo voci anonime comprendenti solo errori fatali; nel file sono presenti sei principali categorie di errore: DEV, MEM, NET, I-O, PRO e OTH; la prima fa riferimento agli errori di dispositivi connessi al sistema, mentre la seconda indica errori sui dispositivi di memoria; la terza categoria comprende tutti gli errori di connessione, la quarta gli errori di lettura e scrittura su storage e la quinta gli errori legati al processore; infine, l'ultima categoria si riferisce a tutti gli altri errori che non appartengono alle categorie precedenti.

7.1.1 Analisi globale

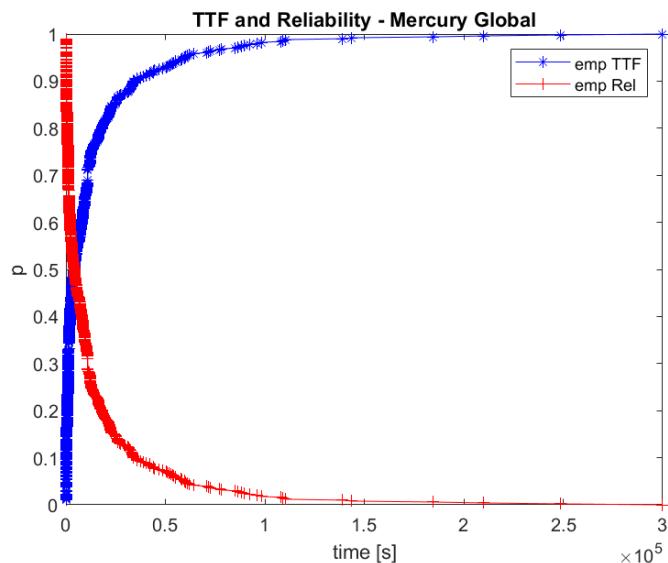
Per effettuare l'analisi FFDA su Mercury sono stati utilizzati gli error logs presenti nel file MercuryErrorLog.txt; prima di tutto è stato calcolato il numero di tuple ottenute al variare della finestra di coalescenza tramite lo script tupleCount_func_CWINpy.sh con l'ausilio del

file tentativo_Cwin.txt in cui sono contenuti i valori di CWIN. Dopo aver ottenuto il numero di tuple per ogni singolo valore di CWIN è stata effettuata una sensitivity analysis, i cui risultati sono mostrati nell'immagine seguente.

10	4283
50	1646
100	839
120	725
190	527
200	508
230	485
290	454
350	440
400	432

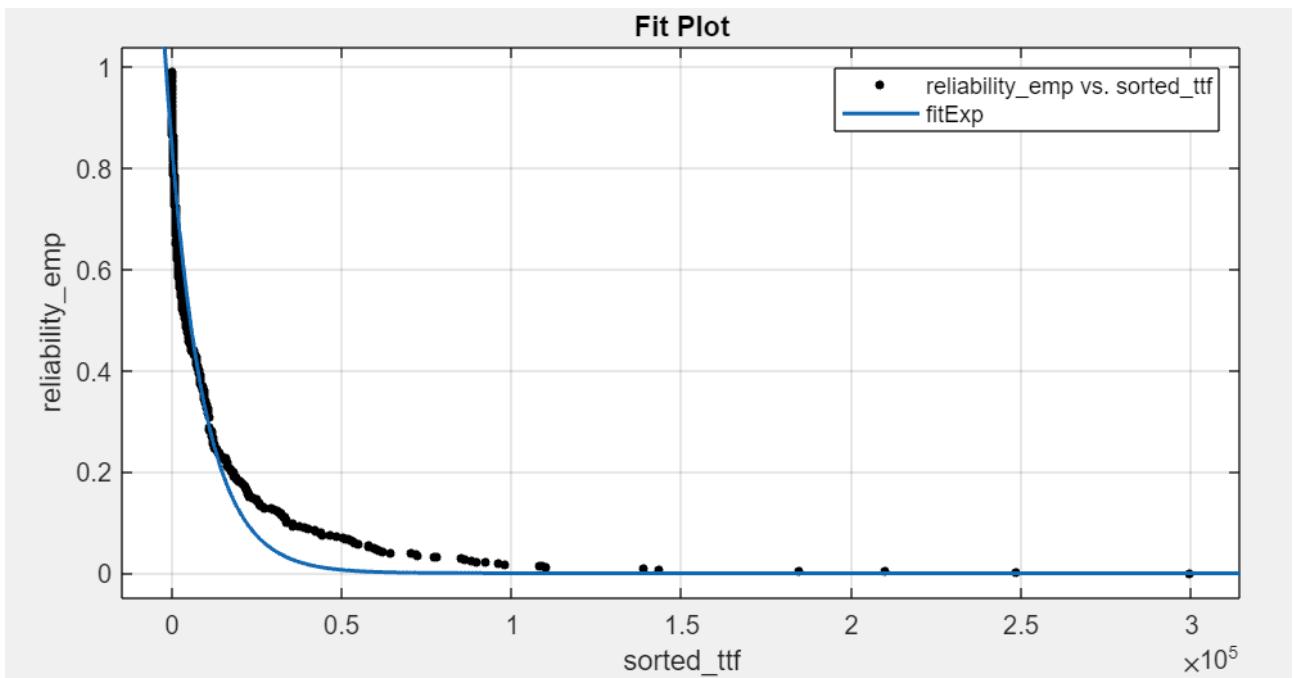


Dall'analisi della curva ottenuta individuiamo che il valore migliore da considerare per la finestra di coalescenza sia 200, posto immediatamente dopo la knee della curva, come per definizione. Utilizzando questo valore della finestra di coalescenza si ottengono 508 tuple. A questo punto, sfruttando la finestra di coalescenza definita precedentemente, è stato effettuato un plot di TTF e Reliability empiriche sfruttando gli interarrivals ottenuti con CWIN pari a 200, dove ogni interarrivals rappresenta un campione del TTF:



A questo punto bisogna andare a capire quale distribuzione teorica rappresenta un buon fit per la reliability empirica; per fare ciò, andiamo ad effettuare il fitting di regressione tramite il tool Curve Fitting di Matlab, considerando tre possibili fit candidati, cioè l'esponenziale semplice, l'esponenziale a due parametri o anche definito iperesponenziale e Weibull. Analizziamo adesso i risultati ottenuti per ogni modello in termini grafici e in funzione di una serie di parametri indicativi della bontà del fit:

1. Esponenziale ($a \cdot e^{b \cdot x}$)

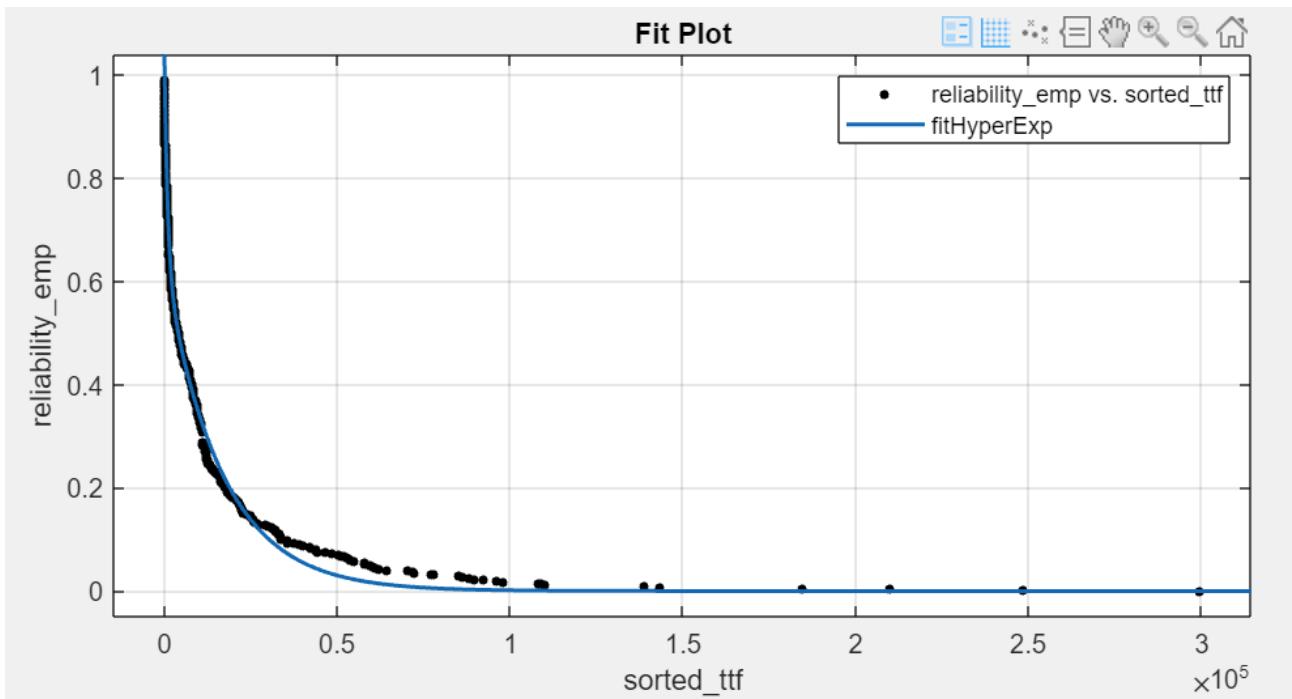


Coefficienti, intervallo di confidenza al 95% e bontà del fit:

Coefficients and 95% Confidence Bounds			
	Value	Lower	Upper
a	0.8390	0.8286	0.8493
b	-0.0001	-0.0001	-0.0001

Goodness of Fit	
	Value
SSE	1.7471
R-square	0.9534
DFE	464.0000
Adj R-sq	0.9533
RMSE	0.0614

2. Iperesponenziale ($a \cdot e^{b \cdot x} + c \cdot e^{d \cdot x}$)

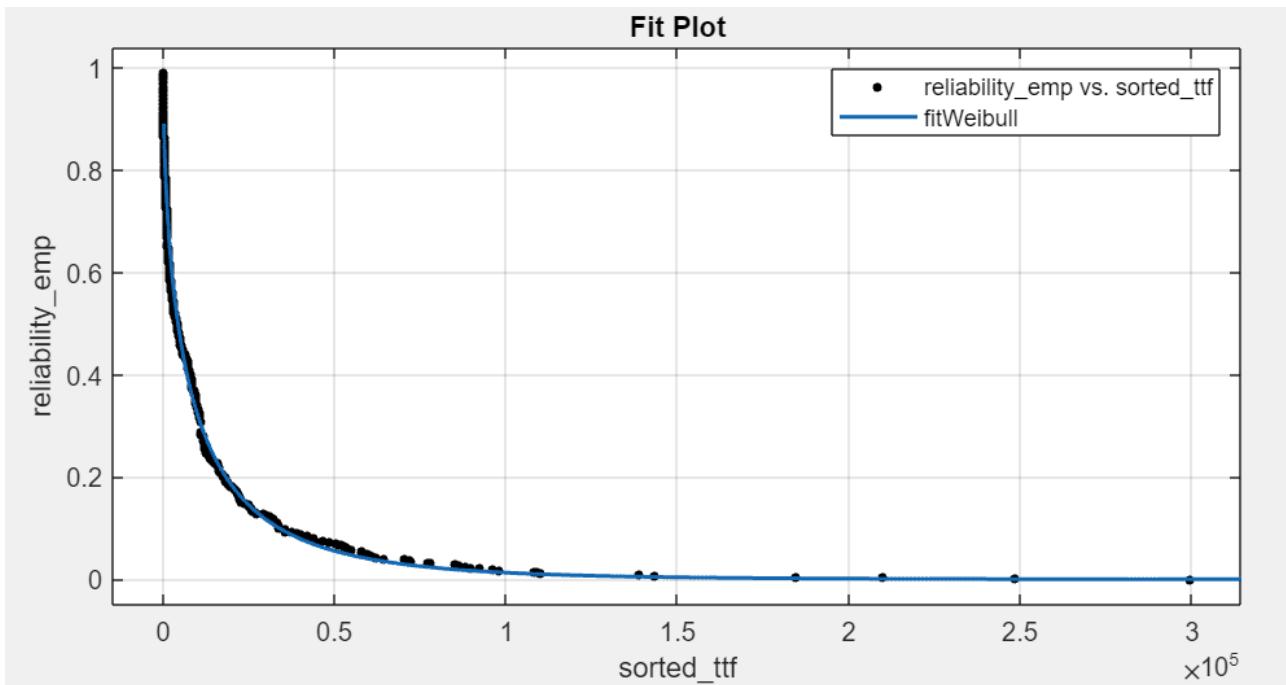


Coefficienti, intervallo di confidenza al 95% e bontà del fit:

Coefficients and 95% Confidence Bounds			
	Value	Lower	Upper
a	0.3922	0.3823	0.4020
b	-0.0010	-0.0011	-0.0010
c	0.6298	0.6207	0.6389
d	-0.0001	-0.0001	-0.0001

Goodness of Fit	
	Value
SSE	0.1457
R-square	0.9961
DFE	462.0000
Adj R-sq	0.9961
RMSE	0.0178

3. Weibull ($e^{-b \cdot x^a}$)



Coefficienti, intervallo di confidenza al 95% e bontà del fit:

Coefficients and 95% Confidence Bounds			
	Value	Lower	Upper
a	0.5823	0.5748	0.5899
b	0.0053	0.0050	0.0056

Goodness of Fit	
	Value
SSE	0.3079
R-square	0.9918
DFE	464.0000
Adj R-sq	0.9918
RMSE	0.0258

Andando ora ad analizzare i parametri di bontà dei tre fit ci rendiamo conto che possiamo tranquillamente escludere l'esponenziale semplice; per quanto invece riguarda gli altri due fit, l'iperesponenziale ha una SSE minore rispetto alla Weibull e un R^2 leggermente maggiore; per cui, andiamo ad effettuare un test di Kolmogorov-Smirnov per poter effettuare la nostra scelta basandoci su un test statistico oltre che visivo; ricordiamo che le ipotesi di questo test sono rispettivamente:

- H_0 : il campione segue la distribuzione specificata
- H_1 : il campione non segue la distribuzione specificata

Tramite Curve Fitting salviamo i fit realizzati nel workspace di Matlab per poter effettuare il test d'ipotesi; per farlo utilizziamo la funzione di libreria kstest2 che prende in input due vettori per vedere se appartengono alla stessa distribuzione continua, e in aggiunta ricorriamo al metodo manuale che permette di verificare se la massima distanza in valore assoluto tra la

distribuzione empirica e teorica D superi o meno un certo valore critico $D_c = \frac{c(\alpha)}{\sqrt{N}}$, dove N è il numero di punti empirici della curva di cui abbiamo effettuato il fit e c(a) è un valore che dipende dalla significatività che stabiliamo. Per a = 0.05 la funzione c(a) è uguale a 1.358. I risultati dei due test utilizzati sono i seguenti:

===== TEST KOLMOGOROV-SMIRNOV (Mercury) =====			
Fit	h_value	p_value	ks2stat_value
Esponenziale	1	0.0000334487	0.1523605150
Iper-Esponenziale	0	0.6665522932	0.0472103004
Weibull	0	0.0724052767	0.0836909871

TEST MANUALE (alpha=0.05, N=466) :

D_c critico = 0.0629

D_exp = 0.1673 X NO
D_hyperexp = 0.0492 ✓ OK
D_weibull = 0.0996 X NO

MIGLIORE FIT: Iper-Esponenziale

A dimostrazione di quanto già affermato precedentemente, l'iperesponenziale risulta essere il fit migliore, in quanto prima di tutto non rigetta l'ipotesi nulla, inoltre ha il p-value più alto, il che indica la massimizzazione della probabilità di non rigettare l'ipotesi nulla, e la distanza massima della reliability empirica, corrispondente al valore della variabile ks2stat, più piccola. Notiamo in realtà che anche Weibull non rigetta l'ipotesi nulla, a differenza di quanto accade con l'esponenziale, ma la prima presenta parametri con valori peggiori rispetto a quelli dell'iperesponenziale. Anche per quanto riguarda il test manuale, notiamo come l'iperesponenziale sia la scelta migliore; infatti, se $D \geq D_c$, il test presuppone un livello di significatività $\geq 95\%$, caso nel quale la reliability teorica non è un buon fit per la reliability empirica. Al contrario, la reliability teorica è un buon fit. Notiamo sempre dall'immagine precedente che l'unico valore di D minore al D critico risulta essere quello dell'iperesponenziale, conferma definitiva del fatto che l'iperesponenziale è la reliability teorica che meglio fitta con quella empirica. Questo risultato ci permette di fare anche una serie di deduzioni sui fallimenti ottenuti, che sembrano essere principalmente legati all'hardware. Approssimando la reliability empirica con la reliability data dall'iperesponenziale, quindi:

$$R(t) = 0.3922 \cdot e^{-0.001t} + 0.6298 \cdot e^{-0.0001t}, t \geq 0$$

Possiamo analizzare l'MTTF utilizzando uno script Matlab, tramite il quale possiamo ottenere i seguenti risultati:

===== MTTF Mercury =====

Coefficienti iper-esponenziale:

a = 0.3922

b = -0.001000

c = 0.6298

d = -0.000100

Reliability teorica:

$$R(t) = 0.3922 \cdot \exp(-0.001000 \cdot t) + 0.6298 \cdot \exp(-0.000100 \cdot t)$$

MTTF = -a/b - c/d

MTTF = 6690.20 secondi

MTTF = 111.50 minuti

MTTF = 1.86 ore

=====

Dopo aver concluso l'analisi globale del sistema, siamo poi passati a un'analisi più specifica basandoci sulle categorie e sui nodi definiti da Mercury; prima di tutto, abbiamo generato le statistiche dei log secondo i due criteri considerati tramite lo script logStatistics.sh, per poi andare ad eseguire la sensitivity analysis prima delle varie categorie e poi dei nodi.

```

== Total error entries ==
80854

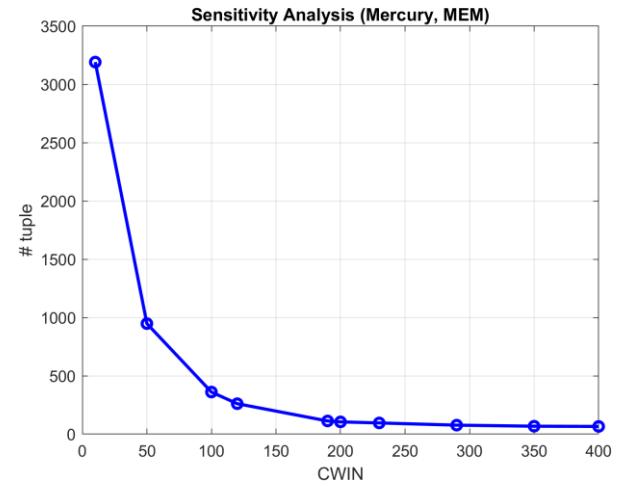
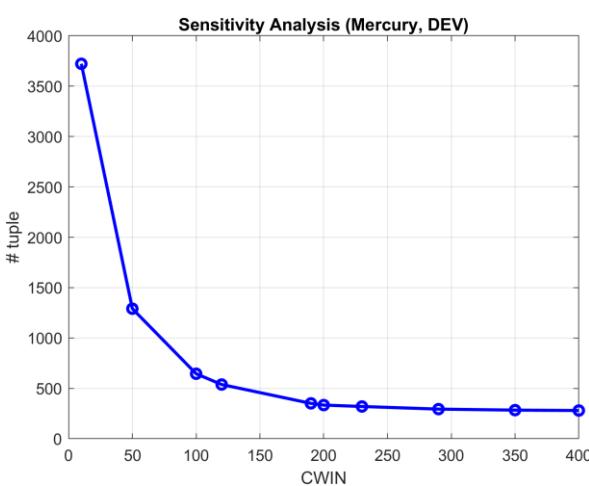
== Breakup by CATEGORY ==
DEV 57248
MEM 12819
I-O 5547
NET 3702
PRO 1504
OTH 34

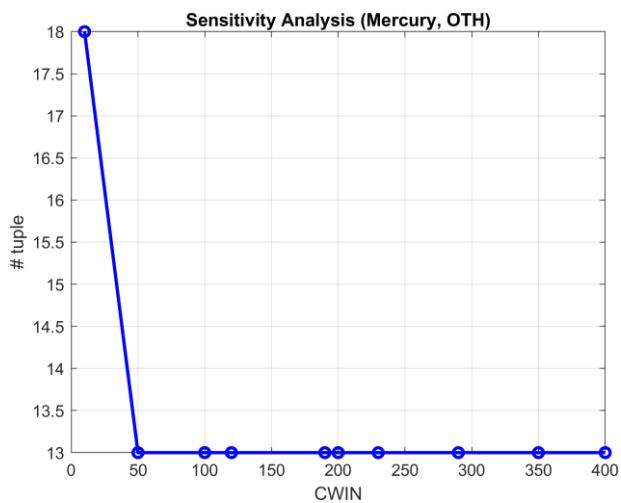
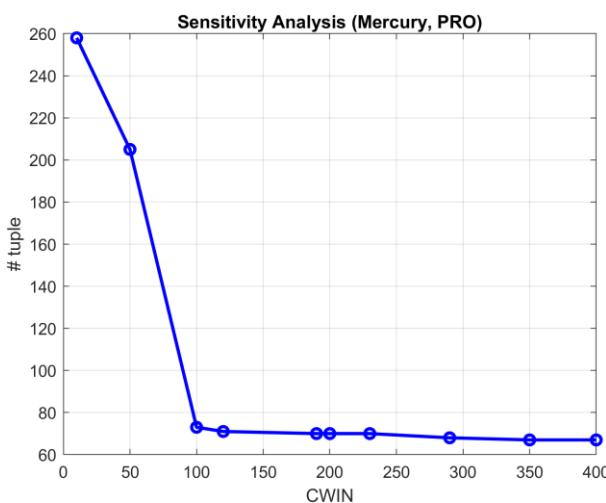
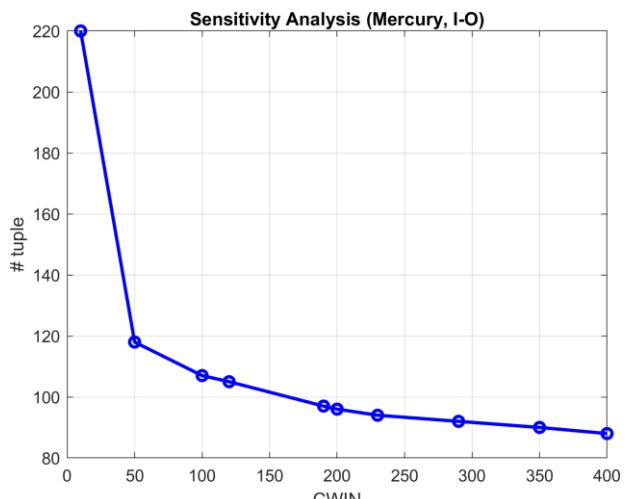
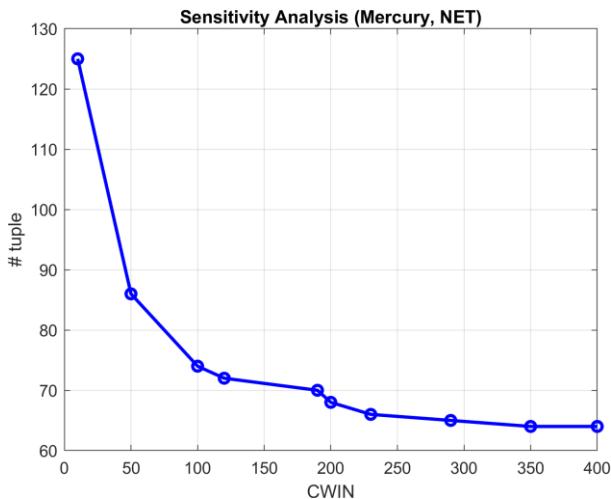
== Breakup by NODE* ==
tg-c401 62340
tg-master 4098
tg-c572 4030
tg-s044 3224
tg-c238 1273
tg-c242 1067
tg-c648 643
tg-login3 382
tg-c117 268
tg-c669 267
* only the 10 most occurring nodes are reported

```

7.1.2 Categorie

Dopo aver individuato le categorie di errore, andiamo a filtrare in base ad esse i file degli error logs tramite lo script filter.sh; per ognuna delle categorie calcoliamo quindi il numero di tuple al variare della finestra di coalescenza, e in seguito effettuiamo la sensitivity analysis per ognuna delle sei categorie a nostra disposizione

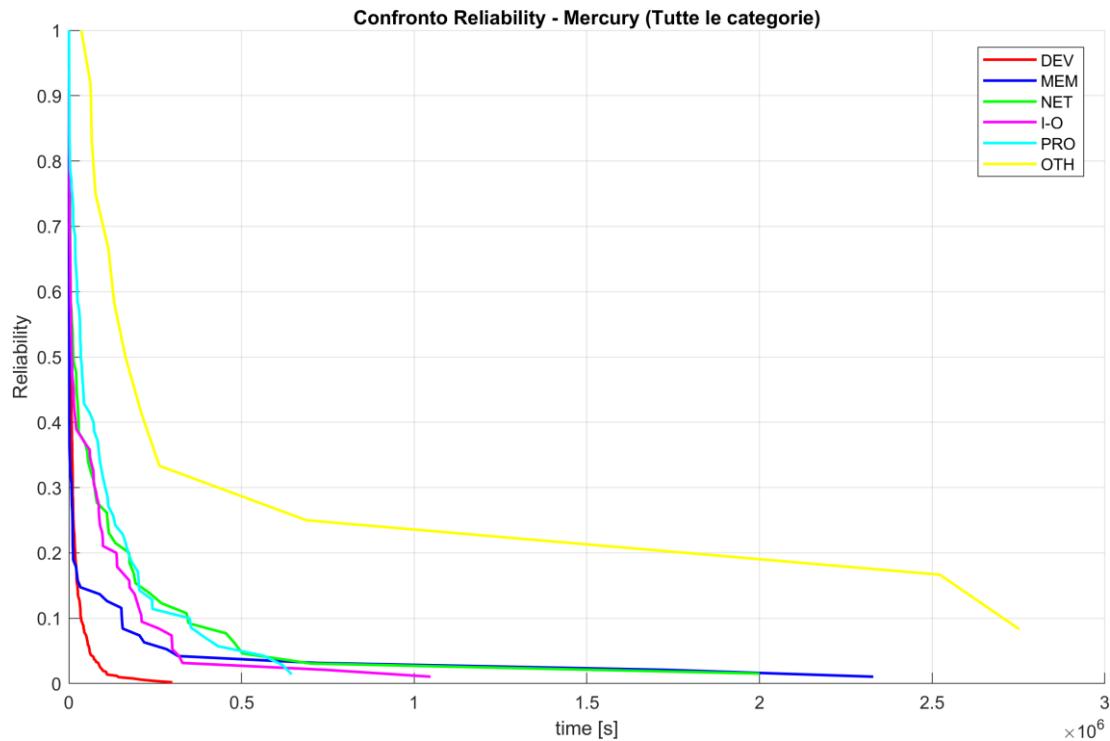




Così come fatto per l'analisi globale, anche in questo caso andiamo a considerare come valore di CWIN per l'analisi della reliability quello immediatamente dopo il ginocchio della curva; è importante notare che il valore mediano è 200, il quale però risulta essere assegnato solo a due categorie su sei; quindi, la CWIN globale considerata precedentemente non è la scelta migliore per ognuna delle categorie.

Categoria	CWIN
DEV	200
MEM	230
I-O	200
NET	230
PRO	100
OTH	100

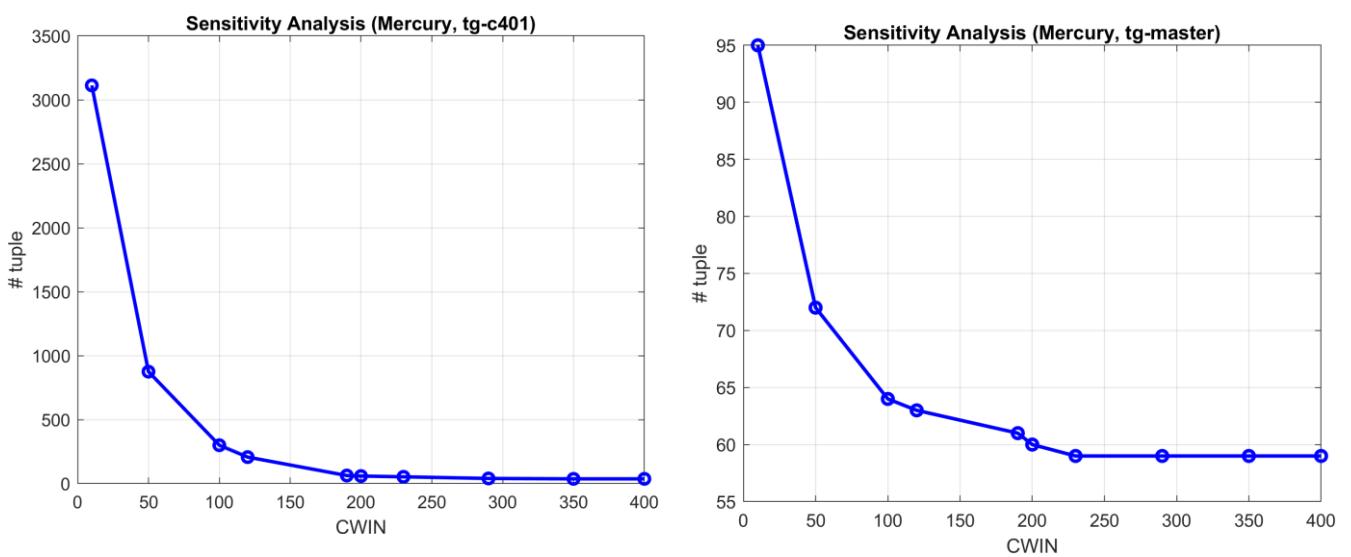
Utilizzando i valori definiti nella tabella precedente abbiamo effettuato il tupling per ogni categoria e successivamente svolgiamo il plot delle reliability empiriche:

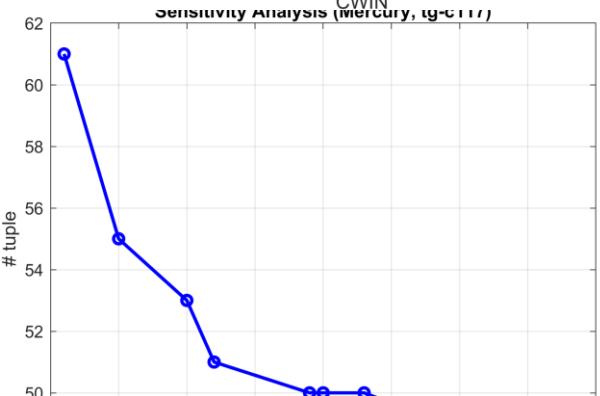
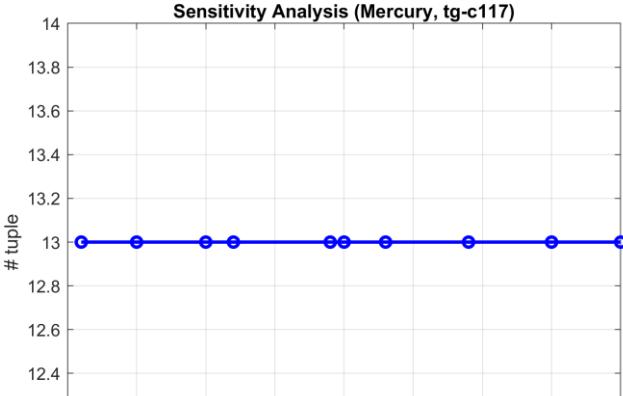
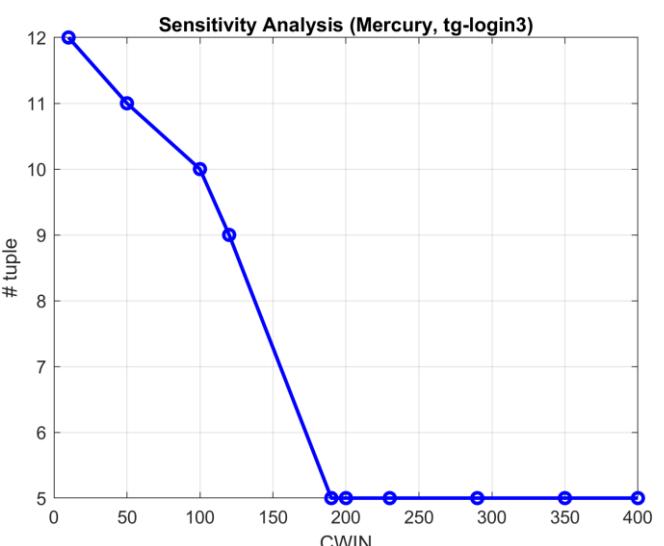
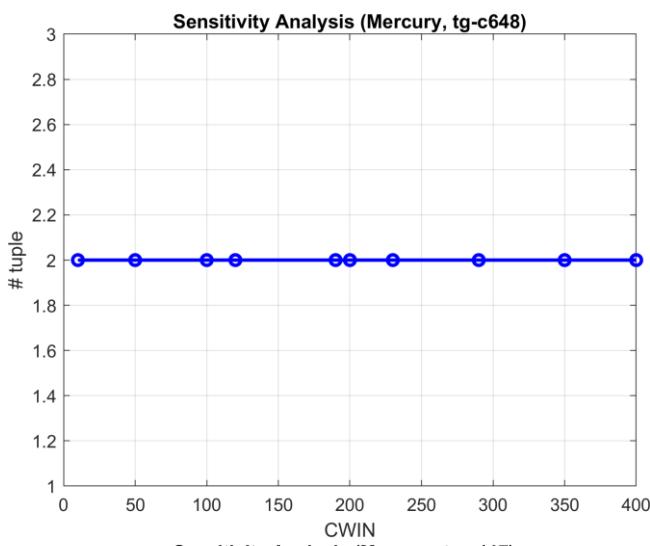
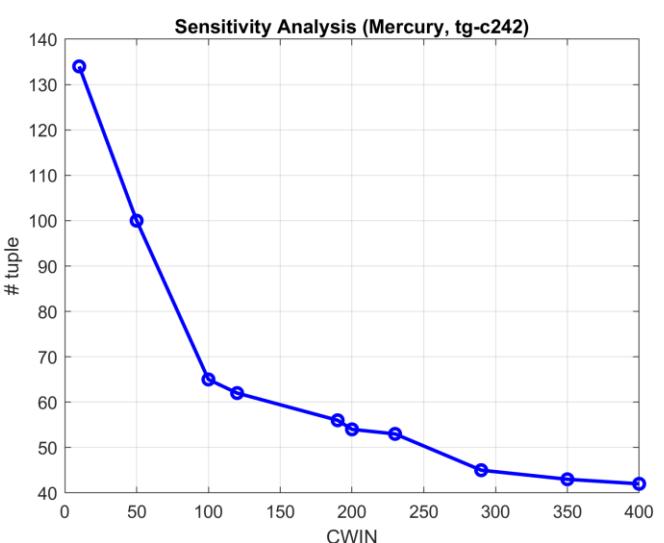
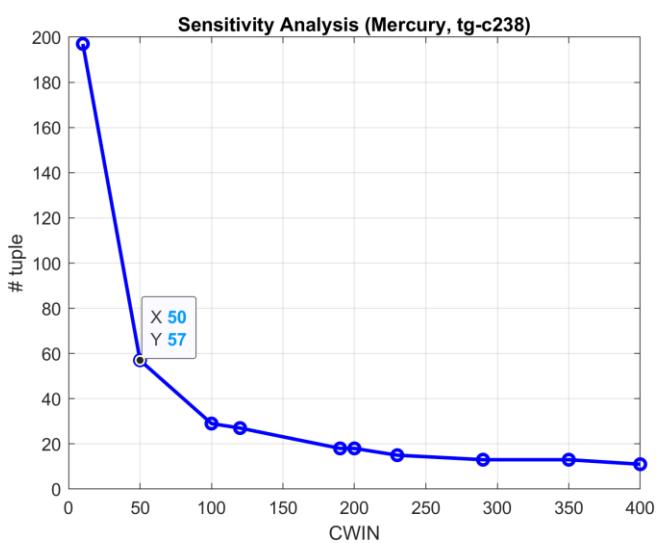
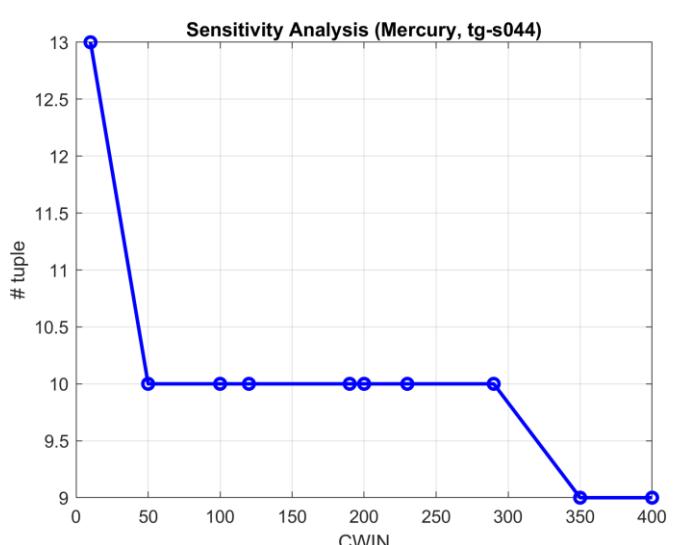
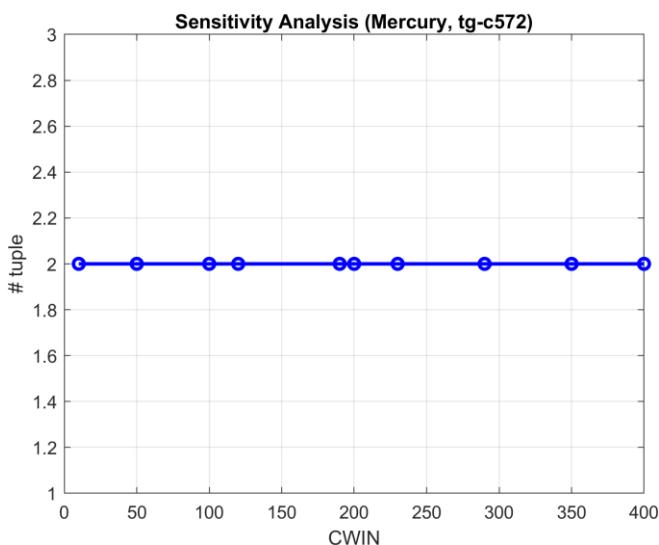


Possiamo notare che la categoria DEV è la più unreliable di tutte, seguita da MEM e I-O; la categoria OTH è invece quella più reliable, mentre PRO e NET hanno un andamento molto simile. Possiamo quindi supporre che la categoria che maggiormente contribuisce ai fallimenti è quella legata agli errori di dispositivi connessi al sistema, la quale può essere considerata il bottleneck del sistema.

7.1.3 Nodi

In maniera analoga a quanto fatto per le categorie filtriemo in base ai nodi Mercury tramite filter.sh, valutiamo il numero di tuple al variare della finestra di coalescenza e realizziamo per i dieci nodi più frequenti la sensitivity analysis:

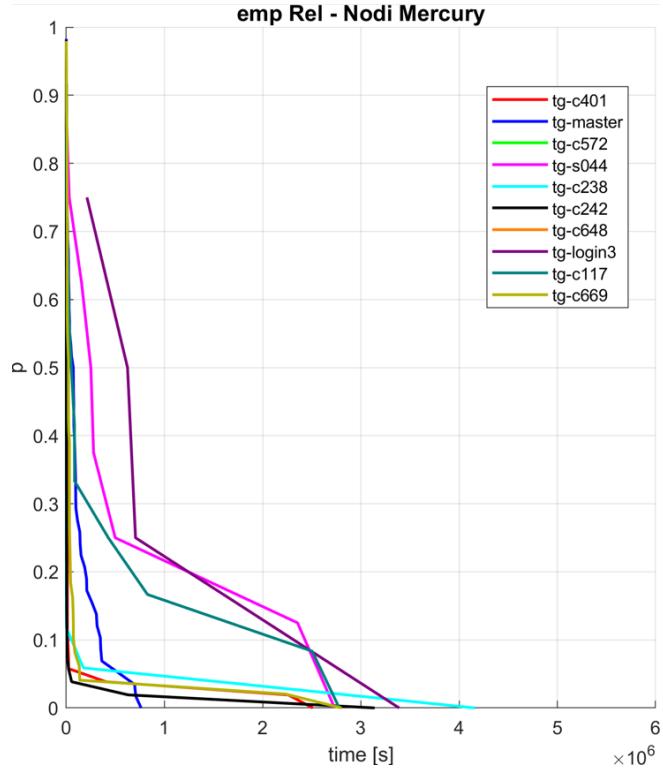




A differenza di quanto accade per le categorie, notiamo che in alcuni dei grafici sui nodi troviamo delle rette costanti; per questi nodi sceglieremo come valore di CWIN un valore casuale, in quanto la scelta è totalmente indifferente; per semplicità, a questi nodi viene assegnato il valore di CWIN pari al valore mediano dei risultati ottenuti, cioè 230.

Nodo	CWIN
tg-c401	230
tg-master	230
tg-c572	230 (indifferente)
tg-s044	350
tg-c238	190
tg-c242	230
tg-c648	230 (indifferente)
tg-login3	200
tg-c117	230 (indifferente)
tg-c669	190

Dopo aver effettuato il tupling per ognuno dei dieci nodi abbiamo plottato le reliability empiriche per effettuare un confronto tra i vari nodi, ottenendo il seguente risultato:



Notiamo in questo caso che sono presenti alcuni nodi particolarmente unreliable, che corrispondono ai nodi c242, c669, c238, c401 e c572; possiamo osservare inoltre che la reliability del nodo master va a zero prima del milione di secondi; quindi, ipotizziamo che la propagazione di fallimenti a partire da più nodi renda anche il nodo tg-master che li gestisce particolarmente unreliable. In generale, i nodi più unreliable sono sicuramente quelli di calcolo, caratterizzati molto probabilmente da un carico di elaborazioni più pesante e dunque sono stressati maggiormente; i nodi di storage e login, unici all'interno della lista dei dieci nodi più frequenti, sono quelli più reliable. A questo punto possiamo andare ad analizzare per i nodi più unreliable le categorie di errori più frequenti:

```
$ bash LogStatistics.sh MercuryErrorLog-tg-c242.txt
== Total error entries ==
1067
== Breakup by CATEGORY ==
DEV 918
MEM 149

$ bash LogStatistics.sh MercuryErrorLog-tg-c238.txt
== Total error entries ==
1273
== Breakup by CATEGORY ==
DEV 1071
MEM 197
NET 3
I-O 2

$ bash LogStatistics.sh MercuryErrorLog-tg-c572.txt
== Total error entries ==
4030
== Breakup by CATEGORY ==
DEV 3176
MEM 845
I-O 9
```

```
$ bash LogStatistics.sh MercuryErrorLog-tg-c669.txt
== Total error entries ==
267
== Breakup by CATEGORY ==
DEV 257
MEM 10

$ bash LogStatistics.sh MercuryErrorLog-tg-c401.txt
== Total error entries ==
62340
== Breakup by CATEGORY ==
DEV 50782
MEM 11558

$ bash LogStatistics.sh MercuryErrorLog-tg-master.txt
== Total error entries ==
4098
== Breakup by CATEGORY ==
NET 3639
I-O 452
OTH 3
DEV 3
MEM 1
```

Notiamo allora che per quanto riguarda i nodi di calcolo gli errori sono principalmente dovuti ai dispositivi connessi (DEV) e alla memoria (MEM); il nodo master presenta invece un numero molto elevato di errori sulla rete (NET), dovuti al fallimento degli altri nodi gestiti dal master come già ipotizzato in precedenza.

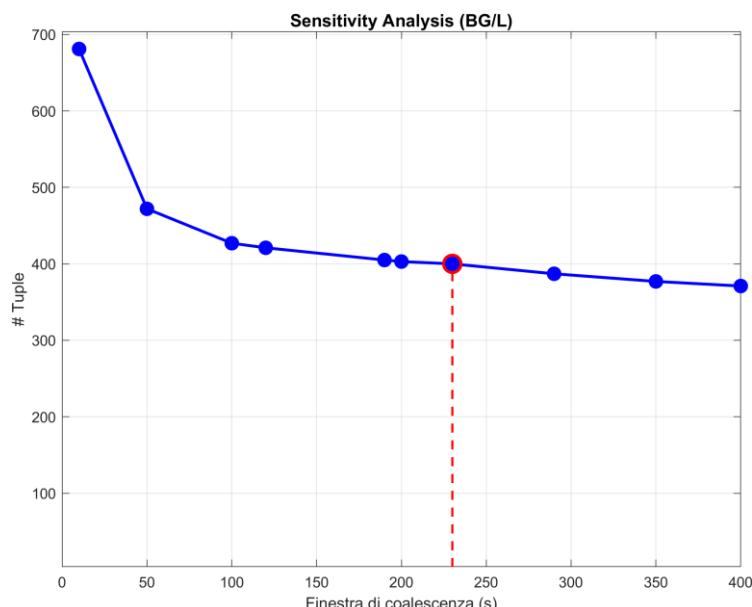
7.2 BG/L

Dopo aver concluso l'analisi di Mercury, prendiamo in considerazione un secondo cluster, BG/L, molto più grande del primo e organizzato in maniera differente: abbiamo una serie di racks, dove ognuno di essi è costituito da midplanes all'interno dei quali sono presenti i nodi; in ogni midplane sono presenti 16 compute cards per nodo, da J02 a J17, mentre i nodi N0, N4, N8 e NC hanno una card I/O aggiuntiva, cioè J18; a sua volta ogni compute card di compone di due compute chips. Per quanto riguarda la nomenclatura, per fare un esempio, con R63-M0-N2-C:J11-U11 indichiamo l'unità 11 della compute card 11, posta all'interno del nodo 2, che a sua volta si trova nel midplane 0 del rack 63. Esattamente come visto per Mercury anche per BG/L abbiamo delle voci di log filtrate, comprendenti solo errori fatali, e ogni voce non riporta una categoria di errore ma un timestamp, il nodo e la card a cui si riferisce la entry, oltre a un messaggio di testo libero.

7.2.1 Analisi globale

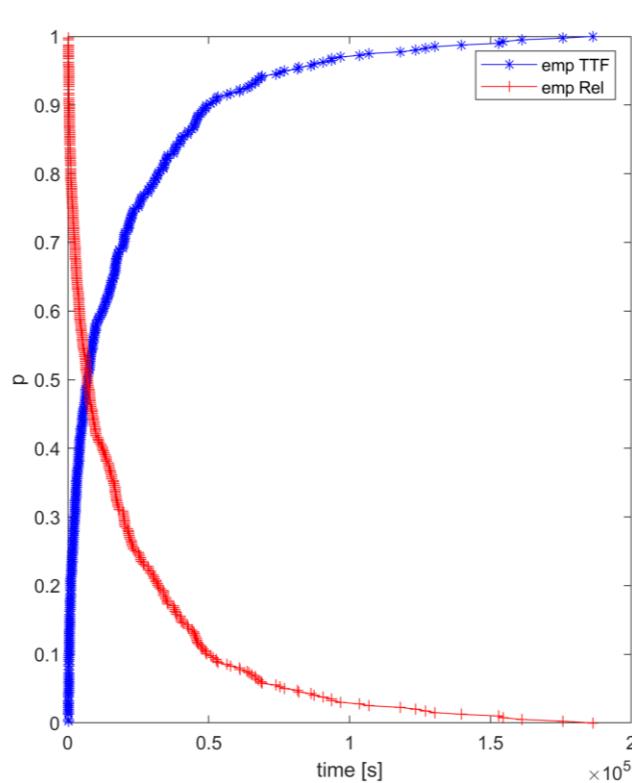
Le operazioni effettuate per BG/L sono esattamente analoghe a quelle effettuate precedentemente per Mercury; prendiamo quindi i log di errore dal file BGLErrorLog.txt, e a partire da esso, sfruttando il file tentative-Cwin.sh, possiamo ricavare il numero di tuple per finestra di coalescenza e successivamente andare ad effettuare la sensitivity analysis, ottenendo il seguente risultato:

10	681
50	472
100	427
120	421
190	405
200	403
230	400
290	387
350	377
400	371



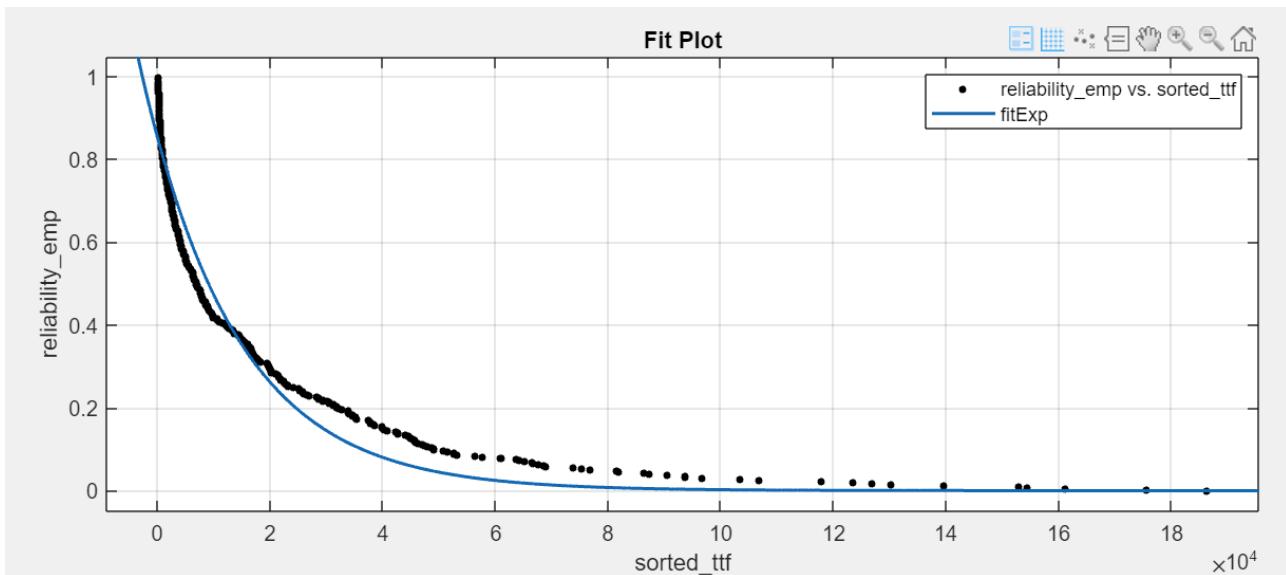
Tramite la knee rule possiamo determinare la finestra di coalescenza scegliendo il punto in cui termina il ginocchio, che nel caso considerato è pari a 230 secondi, ottenendo quindi 400 tuple. Per quanto riguarda il numero di tuple, infatti, dopo 230 secondi la variazione di questo valore è abbastanza bassa.

A questo punto andiamo a plottare TTF e Reliability empiriche, sfruttando gli interarrivals ottenuti con la CWIN scelta precedentemente:



Dopo aver completato questa analisi, possiamo passare all'analisi del miglior fit da utilizzare, realizzata in maniera analogia a quanto visto con Mercury sfruttando il tool Curve Fitting messo a disposizione da Matlab; i tre fit studiati sono gli stessi visti nel caso precedente, cioè esponenziale, iperesponenziale e Weibull.

1. *Esponenziale ($a \cdot e^{b \cdot x}$)*



Coefficienti, intervallo di confidenza al 95% e bontà del fit:

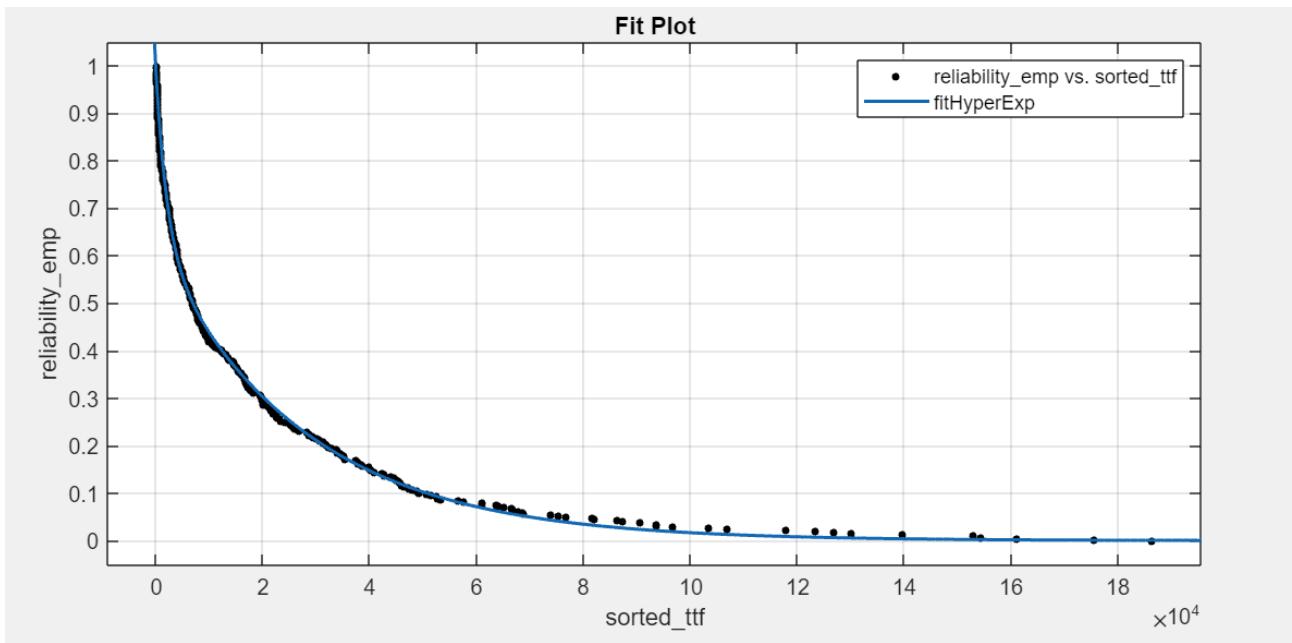
Coefficients and 95% Confidence Bounds

	Value	Lower	Upper
a	0.8533	0.8423	0.8643
b	-0.0001	-0.0001	-0.0001

Goodness of Fit

	Value
SSE	1.3470
R-square	0.9576
DFE	389.0000
Adj R-sq	0.9575
RMSE	0.0588

2. Iperesponenziale ($a \cdot e^{b \cdot x} + c \cdot e^{d \cdot x}$)



Coefficienti, intervallo di confidenza al 95% e bontà del fit:

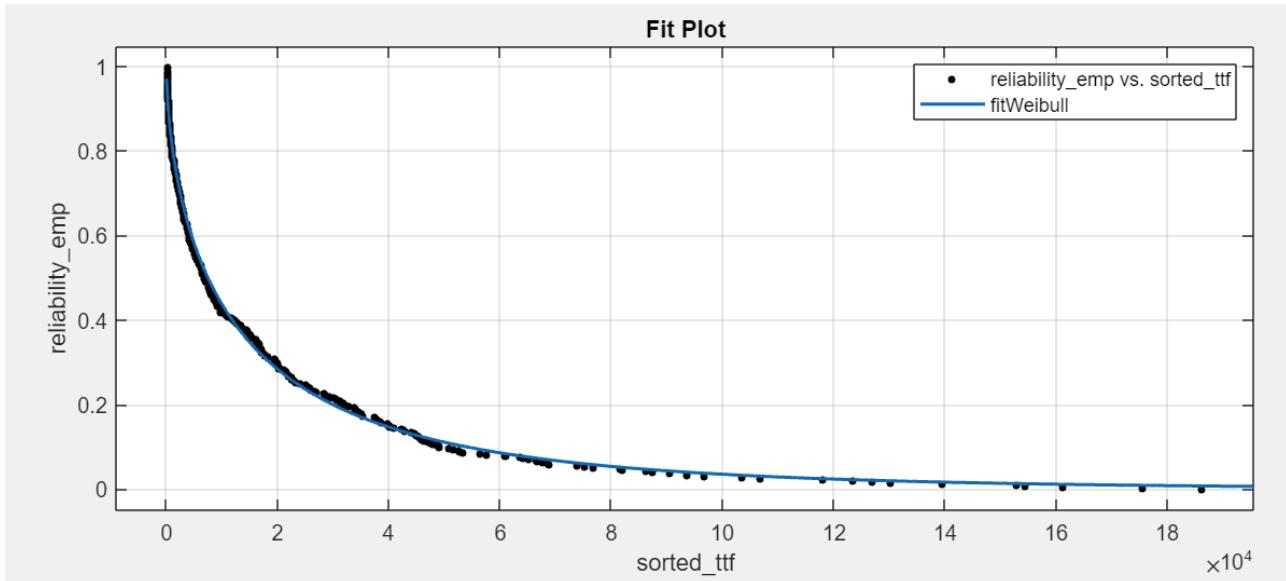
Coefficients and 95% Confidence Bounds

	Value	Lower	Upper
a	0.3682	0.3603	0.3761
b	-0.0005	-0.0005	-0.0004
c	0.6233	0.6150	0.6315
d	-0.0000	-0.0000	-0.0000

Goodness of Fit

	Value
SSE	0.0634
R-square	0.9980
DFE	387.0000
Adj R-sq	0.9980
RMSE	0.0128

3. Weibull ($e^{-b \cdot x^a}$)



Coefficienti, intervallo di confidenza al 95% e bontà del fit:

Coefficients and 95% Confidence Bounds

	Value	Lower	Upper
a	0.6067	0.6004	0.6130
b	0.0031	0.0029	0.0033

Goodness of Fit

	Value
SSE	0.1393
R-square	0.9956
DFE	389.0000
Adj R-sq	0.9956
RMSE	0.0189

Così come fatto per Mercury, possiamo renderci conto del fatto che ancora una volta il fit migliore sia in termini di SSE che in termini di R^2 è quello iperesponeziale; a questo punto effettuiamo il test d'ipotesi di Kolmogorov-Smirnov sfruttando ancora la funzione kstest2 e il test manuale. I risultati ottenuti sono quelli mostrati nell'immagine successiva:

```
===== TEST KOLMOGOROV-SMIRNOV (BG/L) =====
Fit          h_value  p_value      ks2stat_value
Esponenziale    1        0.0004226393  0.1457800512
Iper-Esponenziale 0        0.8459487706  0.0434782609
Weibull        0        0.2985182542  0.0690537084
=====
```

TEST MANUALE (alpha=0.05, N=391):

D_c critico = 0.0687

D_exp	= 0.1561	X NO
D_hyperexp	= 0.0499	✓ OK
D_weibull	= 0.0794	X NO

MIGLIORE FIT: Iper-Esponenziale

L'esponenziale, come accade precedentemente, rigetta l'ipotesi nulla, mentre l'iperesponenziale e Weibull no; inoltre, però l'iperesponenziale presenta il p-value più alto e la ks2stat più bassa, quindi è migliore della Weibull. Anche tramite il test manuale arriviamo alle stesse conclusioni, in quanto solo l'iperesponenziale ha una distanza massima in valore assoluto minore del valore critico ad un livello di significatività del 95%, per cui possiamo confermare quanto affermato finora. A questo punto, andiamo a calcolare l'MTTF; prima di tutto approssimiamo la reliability empirica con la reliability teorica data dall'iperesponenziale, cioè:

$$R(t) = 0.3682 \cdot e^{-0.000468t} + 0.6232 \cdot e^{-3.606 \cdot 10^{-5}t}$$

A questo punto andiamo a valutare l'MTTF sfruttando lo stesso script Matlab utilizzato per Mercury ottenendo:

```
===== MTTF BG/L =====
Coeffienti iper-esponenziale:
  a = 0.3682
  b = -0.000468
  c = 0.6233
  d = -0.000036

Reliability teorica:
  R(t) = 0.3682*exp(-0.000468*t) + 0.6233*exp(-0.000036*t)

  MTTF = -a/b - c/d
  MTTF = 18071.83 secondi
  MTTF = 301.20 minuti
  MTTF = 5.02 ore
=====
```

Concludiamo così l'analisi globale del cluster BG/L; a questo punto, analogamente a quanto fatto con Mercury, bisogna andare ad effettuare l'analisi delle cards e dei nodi, limitandoci a considerare le 10 cards e i 10 nodi più frequenti, individuate tramite lo script logStatistics.sh:

```

== Total error entries ==
125624

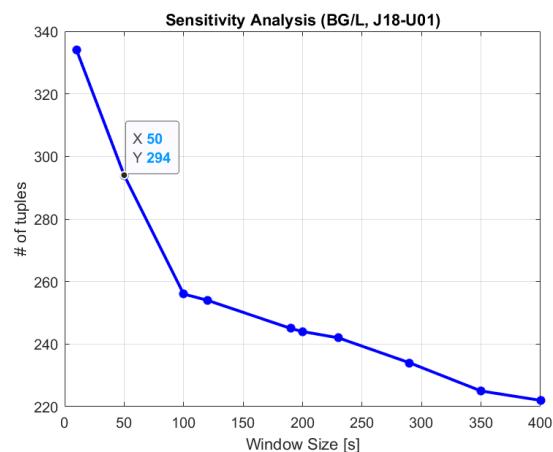
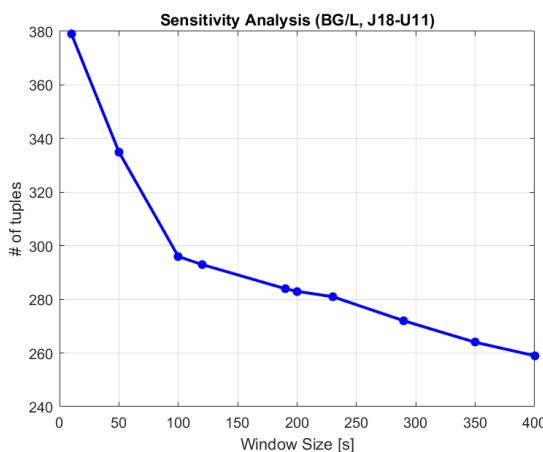
== Breakup by CATEGORY ==
j18-u11 50055
j18-u01 49932
j14-u01 2257
j12-u01 1877
j07-u01 1780
j10-u11 1333
j03-u11 1020
j16-u11 973
j06-u11 960
j11-u11 888
j17-u11 824
j09-u01 808
j16-u01 753
j09-u11 746
j08-u01 741
j03-u01 701
j11-u01 700
j05-u11 670
j04-u01 655
j13-u01 647
j15-u01 633
j17-u01 602
j12-u11 596
j04-u11 593
j13-u11 563
j08-u11 560
j10-u01 554
j02-u01 524
j05-u01 496
j15-u11 454
j02-u11 449
j14-u11 445
j07-u11 445
j06-u01 430

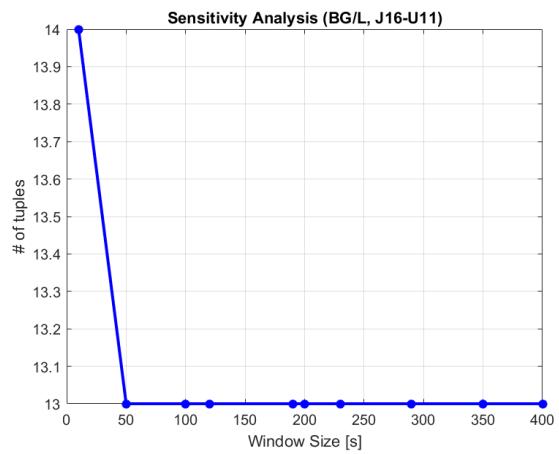
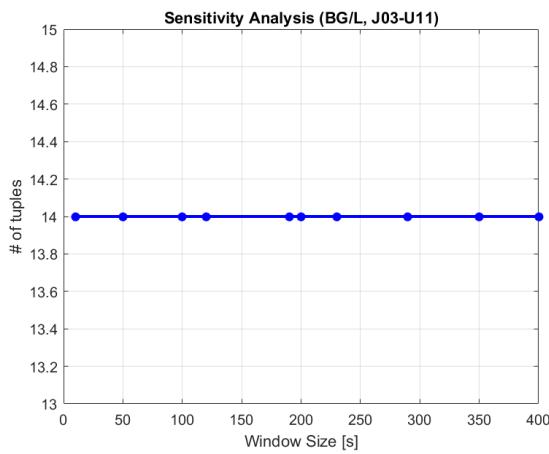
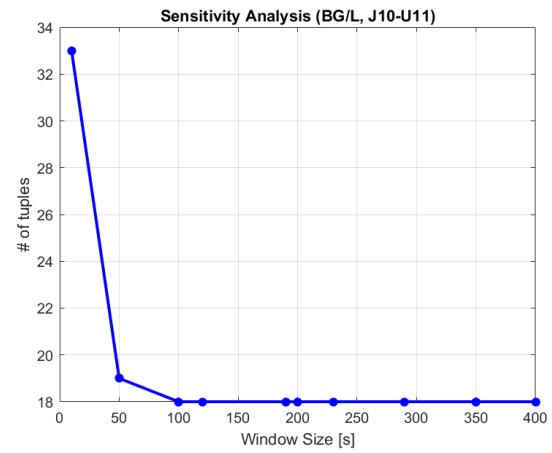
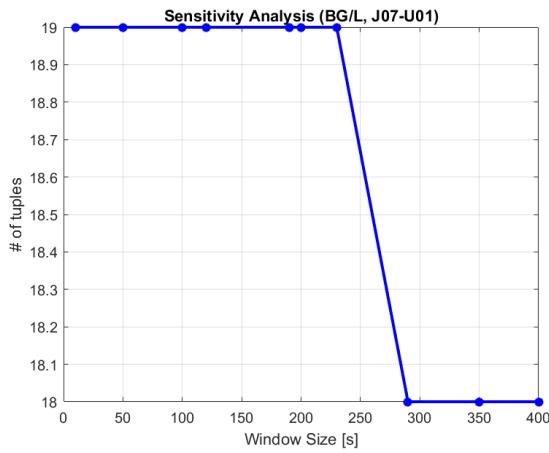
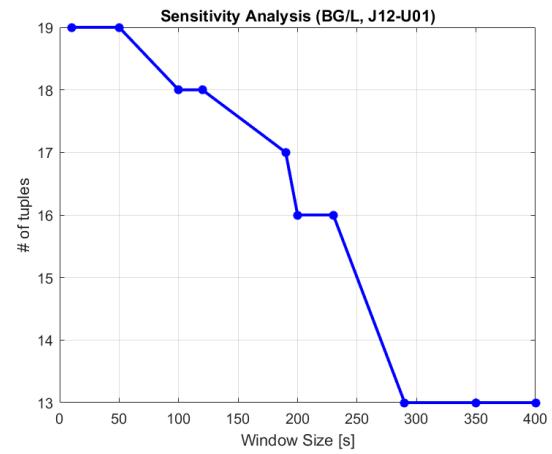
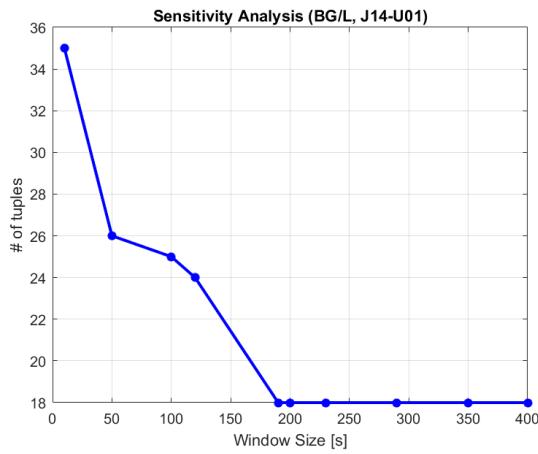
== Breakup by NODE* ==
R71-M0-N4 1716
R12-M0-N0 1563
R63-M0-N2 976
R03-M1-NF 960
R63-M0-N0 791
R36-M1-N0 788
R62-M0-N4 515
R63-M0-NC 460
R63-M0-N8 454
R63-M0-N4 452
* only the 10 most occurring nodes are reported

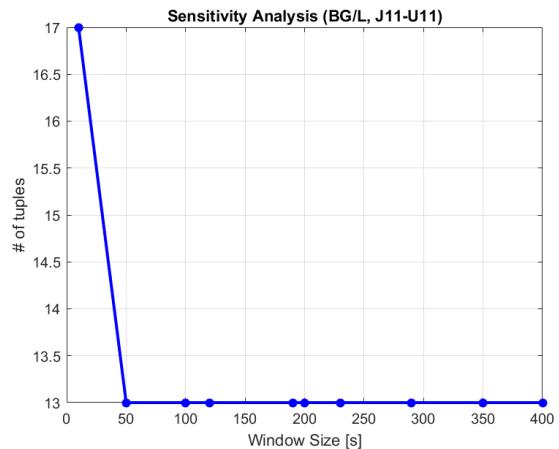
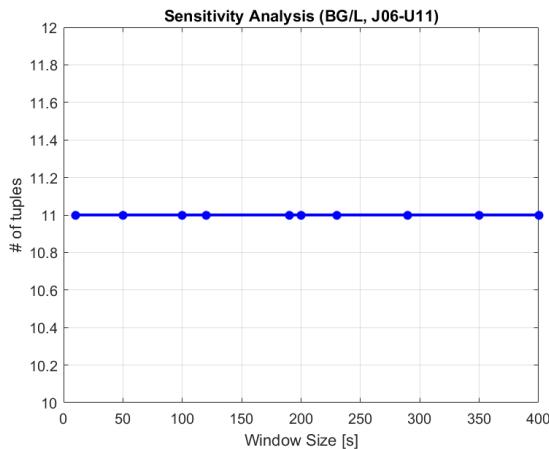
```

7.2.2 Cards

Adesso, per quanto riguarda le cards, abbiamo come detto precedentemente di concentrare la nostra attenzione sulle dieci cards con più voci di errore ottenute tramite lo script logStatistics.sh. Andando ad analizzare il numero di tuple per finestra di coalescenza per ogni singola card, possiamo andare a determinare la CWIN da utilizzare per ognuna di esse tramite la sensitivity analysis, la quale ci fornisce i seguenti risultati:



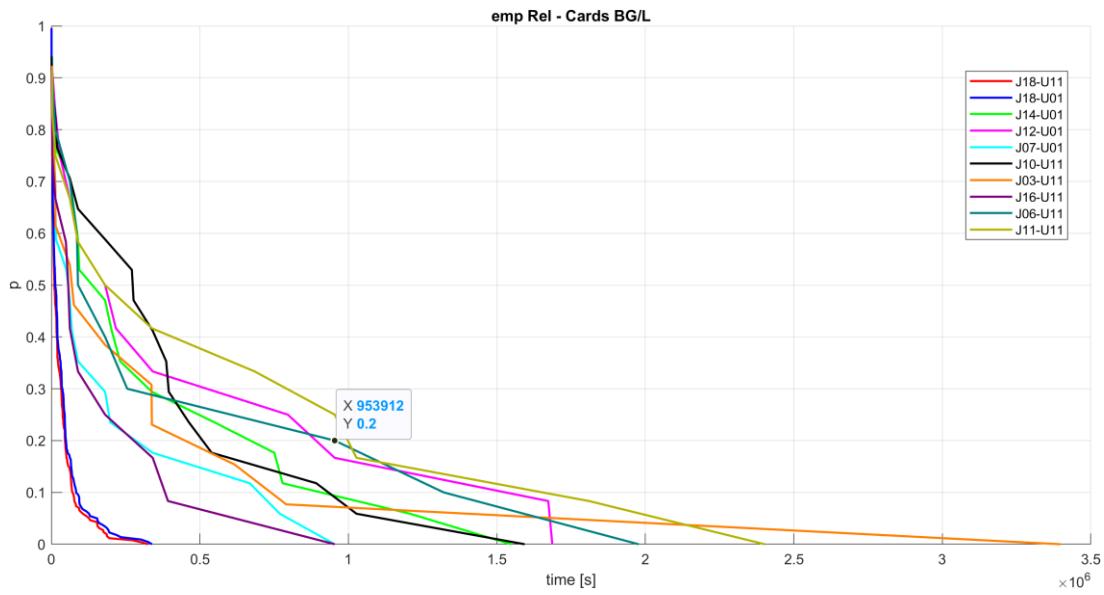




Anche in questo caso, ovviamente, sceglieremo come valore di CWIN quello immediatamente successivo al ginocchio della curva; per le due cards il cui grafico rappresenta una retta, così come fatto per i nodi di Mercury, utilizziamo il valore mediano, cioè 230, abbastanza vicino al valore della CWIN globale. I valori utilizzati sono mostrati nella seguente tabella:

Card	CWIN
J18-U11	400
J18-U01	400
J14-U01	150
J12-U01	350
J07-U01	350
J10-U11	100
J03-U11	230 (indifferente)
J16-U11	50
J06-U11	230 (indifferente)
J11-U11	100

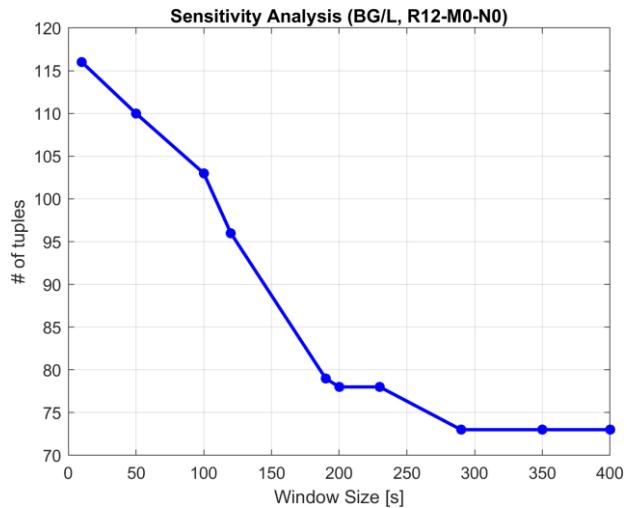
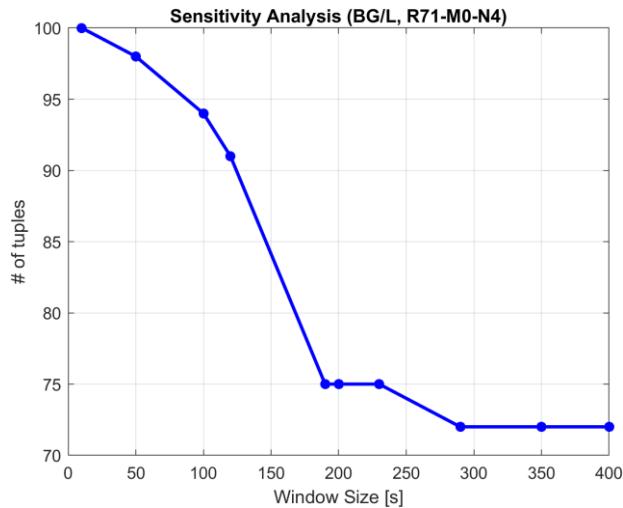
Utilizzando questi valori possiamo andare ad effettuare il tupling per ogni singola card per poi andare a plottare le reliability empiriche, ottenendo i seguenti risultati:

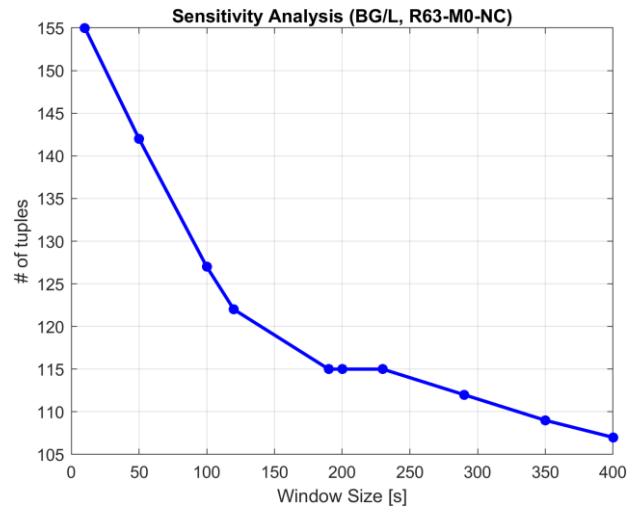
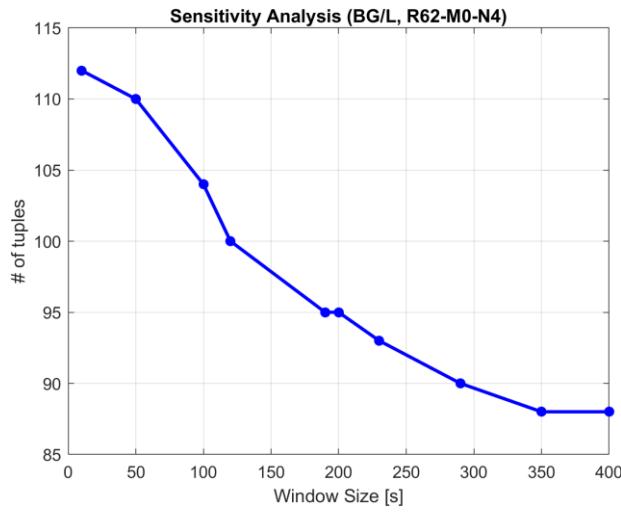
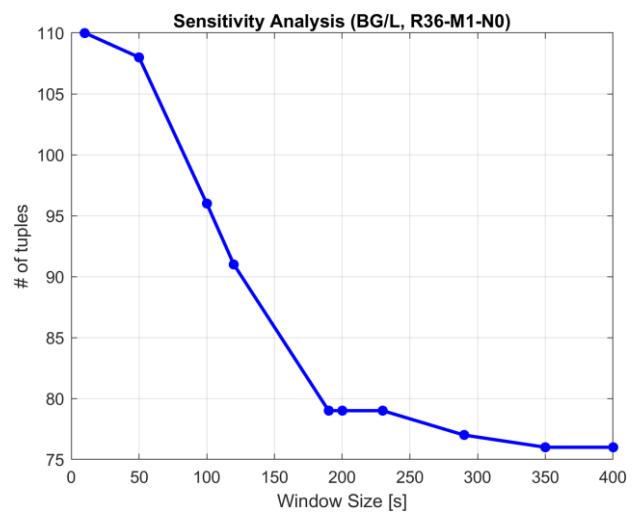
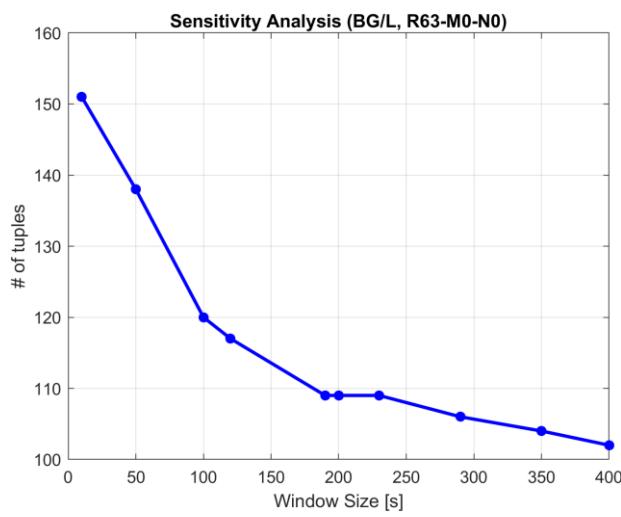
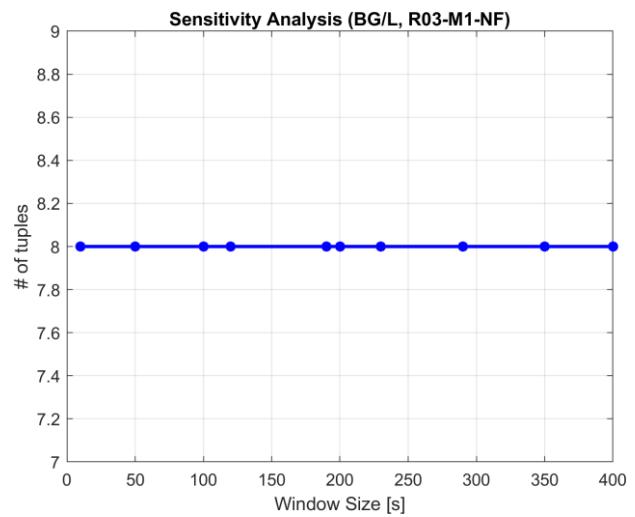
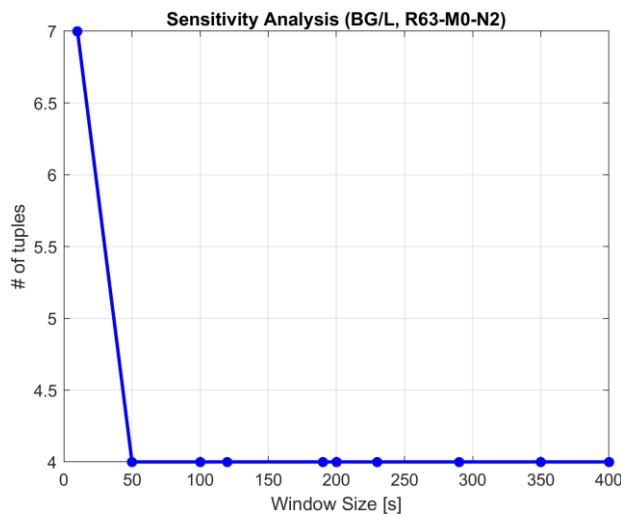


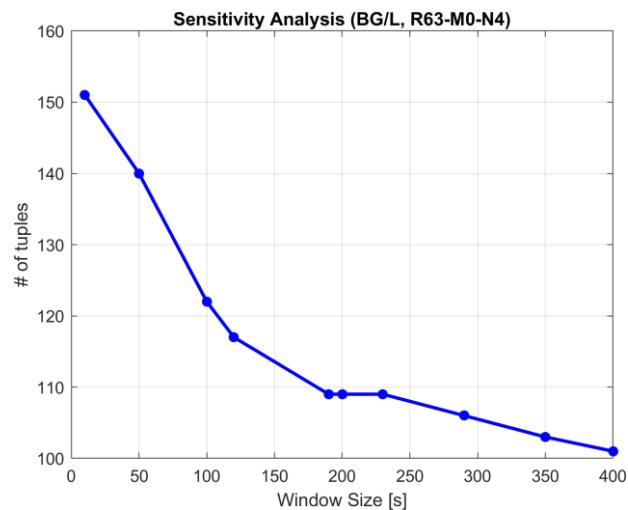
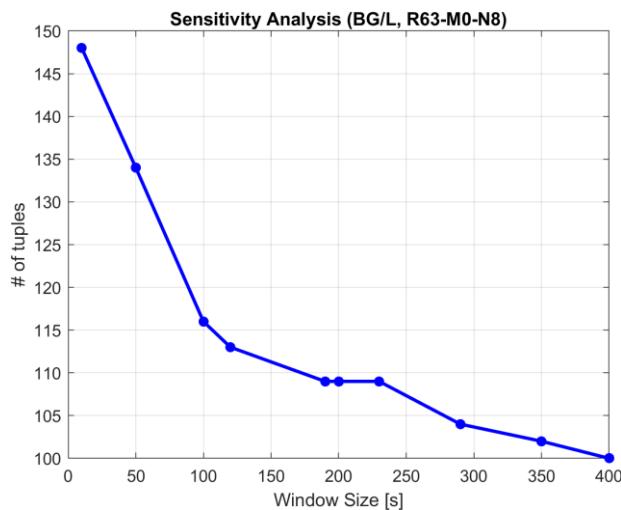
Notiamo quindi che le due cards J18-U11 e J18-U01 sono quelle più unreliable e hanno due curve di reliability molto simili, e possiamo osservare che esse sono le card I/O aggiuntive che non tutti i nodi possiedono. Questo ci permette di ipotizzare che una bottleneck possibile per la dependability del sistema è rappresentato dagli errori I/O.

7.2.3 Nodi

Per concludere l'analisi, andiamo a svolgere sui nodi le stesse operazioni precedentemente svolte sulle cards; andiamo allora a valutare le sensitivity analysis dei nodi più frequenti, ottenendo i seguenti risultati:



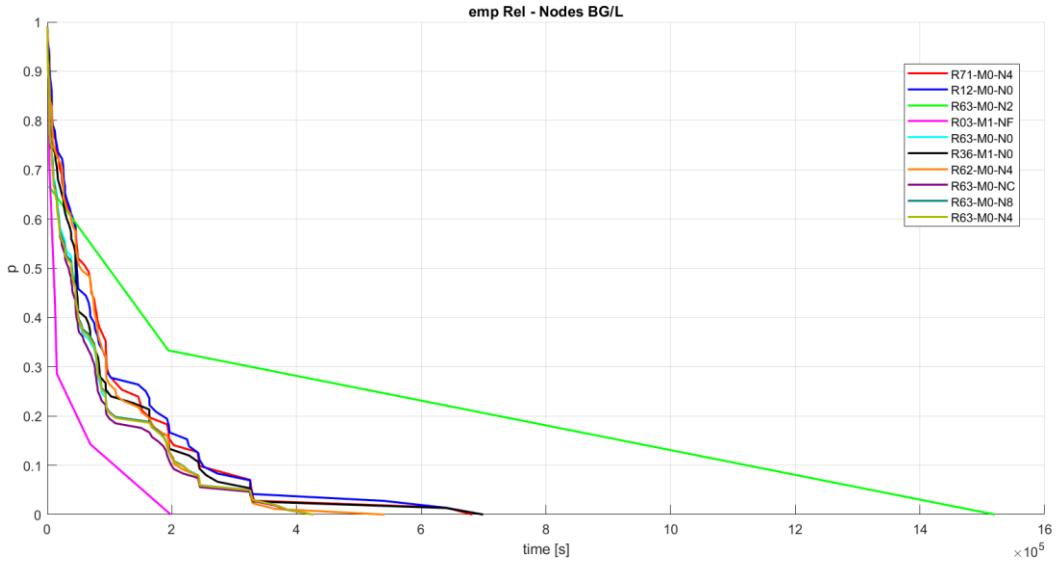




Per quanto riguarda i valori della finestra di coalescenza scelti, andiamo a prendere sempre il valore immediatamente successivo al ginocchio quando possibile; per il solo nodo in cui il grafico rappresenta una retta utilizziamo come CWIN il valore mediano che in questo caso risulta essere 320. Otteniamo allora:

Nodo	CWIN
R71-M0-N4	350
R12-M0-N0	350
R63-M0-N2	50
R03-M1-NF	350 (indifferente)
R63-M0-N0	400
R36-M1-NC	350
R62-M0-N4	350
R63-M0-NC	350
R63-M0-N8	350
R63-M0-N4	350

Abbiamo quindi eseguito il tupling utilizzando i seguenti valori di CWIN e infine sono state plottate le reliability empiriche per ogni singolo nodo, tentando così di verificare se le ipotesi da noi effettuate in precedenza siano valide. I risultati ottenuti sono i seguenti:



Possiamo osservare quindi che il nodo più unreliable è R03-M1-NF, il quale va a zero molto più rapidamente di tutti gli altri nodi, mentre quello più reliable è con ampio margine R63-M0-N2; inoltre possiamo notare come tutti i nodi che presentano la card I/O aggiuntiva hanno delle reliability simili, raffiguranti il loro comportamento quasi identico. Analizzando adesso i report delle statistiche dei nodi dotati di card I/O aggiuntiva:

```
$ bash logStatistics.sh BGLErrorLog-R71-M0-N4.txt
== Total error entries ==
1716
== Breakup by CATEGORY ==
J14-U01 1562
J18-U11 77
J18-U01 77
$ bash logStatistics.sh BGLErrorLog-R63-M0-N0.txt
== Total error entries ==
791
== Breakup by CATEGORY ==
J18-U11 132
J18-U01 131
J11-U11 118
$ bash logStatistics.sh BGLErrorLog-R62-M0-N4.txt
== Total error entries ==
515
== Breakup by CATEGORY ==
J13-U01 261
J18-U11 98
J18-U01 94
$ bash logStatistics.sh BGLErrorLog-R63-M0-N8.txt
== Total error entries ==
454
== Breakup by CATEGORY ==
J18-U11 132
J18-U01 128
J10-U01 8
```

```
$ bash logStatistics.sh BGLErrorLog-R71-M0-N4.txt
== Total error entries ==
1716
== Breakup by CATEGORY ==
J14-U01 1562
J18-U11 77
J18-U01 77
$ bash logStatistics.sh BGLErrorLog-R36-M1-N0.txt
== Total error entries ==
788
== Breakup by CATEGORY ==
J16-U11 585
J18-U11 103
J18-U01 100
$ bash logStatistics.sh BGLErrorLog-R63-M0-NC.txt
== Total error entries ==
460
== Breakup by CATEGORY ==
J18-U11 135
J18-U01 133
J17-U11 6
$ bash logStatistics.sh BGLErrorLog-R63-M0-N4.txt
== Total error entries ==
452
== Breakup by CATEGORY ==
J18-U11 130
J18-U01 130
J17-U11 6
```

Osserviamo che nella top 3 di ogni nodo è sempre presente la card I/O aggiuntiva, la quale sarà sempre responsabile della maggior parte degli errori in ogni nodo in cui è presente; quindi, possiamo pensare che esista un rapporto tra l'unreliability del sistema e queste card aggiuntive; quindi, il bottleneck del sistema è dovuto proprio agli errori di I/O.

7.3 Conclusioni

Per concludere l'analisi FFDA, consideriamo alcune particolarità che abbiamo individuato durante l'intero processo; innanzitutto, abbiamo visto come dall'analisi di categorie e nodi non sia possibile scegliere sempre la stessa finestra di coalescenza tra nodi o categorie differenti, in quanto sono caratterizzati da reliability empiriche in molti casi estremamente differenti tra loro. Quindi la CWIN calcolata per il sistema durante l'analisi globale di entrambi i cluster non risulta essere un valore buono per ogni nodo/categoria. Inoltre, abbiamo osservato come nodi simili da un punto di vista funzionale posseggono in buona parte dei casi delle curve di reliability confrontabili, mentre per quanto riguarda i fallimenti abbiamo visto che esistono per entrambi i cluster delle relazioni tra i fallimenti per categoria e i fallimenti per nodo, soprattutto in BG/L dove gli errori sono dovuti principalmente alla presenza delle cards I/O aggiuntive e ben otto nodi su dieci di quelli che generano più errori sono quelli che presentano al loro interno queste cards. Andando invece ad analizzare i nodi che contribuiscono maggiormente al numero di fallimenti totale del sistema abbiamo potuto individuare i potenziali bottleneck per la dependability. Infine, grazie al MTTF, possiamo affermare che in generale BG/L è più reliable di Mercury, in quanto ha un MTTF teorico ben più grande di quello del primo dataset analizzato.