



Università degli Studi di Napoli "Federico II"

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

Anno Accademico 2020/2021

C.d.L. in Ingegneria Informatica

**Elaborato d'esame
in
Impianti di Elaborazione**

prof. Domenico Cotroneo

M63001040 Antimo Iannucci
M63000988 Mario Pace

Indice

1 Es. 1 - Benchmarking	1
1.1 Traccia	1
1.2 Strumenti utilizzati	1
1.3 Introduzione	1
1.4 Analisi del problema	1
1.5 Soluzione	2
1.5.1 Stima della dimensione campionaria	2
1.5.2 Generazione campioni differenza	3
1.5.3 Analisi statistica in JMP	5
1.5.4 Considerazioni	11
2 Es. 2 - Workload Characterization	12
2.1 Traccia	12
2.2 Strumenti utilizzati	12
2.3 Introduzione	12
2.4 Analisi del problema	13
2.5 Soluzione	13
2.5.1 Studio della correlazione	13
2.5.2 Principal Component Analysis	14
2.5.3 Clustering	17
2.5.4 Valutazione del Trade-off	19
2.5.5 Workload sintetico	21

3 Es. 3 - Web Server Performance Analysis	24
3.1 Traccia	24
3.2 Strumenti utilizzati	24
3.3 Workload Characterization	24
3.3.1 Fase 1: Workload Characterization HL e LL	25
3.3.2 Fase 2: Generazione di LL'_c a partire da HL_c	31
3.3.3 Fase 3: Validazione di LL'_c	32
3.4 Capacity Test	36
3.5 Risorsa Small	37
3.6 Risorsa Big	38
3.7 Risorse Random	39
3.8 Design of Experiment (DoE)	40
4 Es. 4 - Reliability	49
4.1 Esercizio 1	49
4.1.1 Traccia	49
4.1.2 Risoluzione	49
4.2 Esercizio 2	53
4.2.1 Traccia	53
4.2.2 Risoluzione	53
4.3 Esercizio 3	55
4.3.1 Traccia	55
4.3.2 Risoluzione	56
4.4 Esercizio 4	59
4.4.1 Traccia	59
4.5 Esercizio 5	64
4.5.1 Traccia	64
4.5.2 Risoluzione (a)	65
4.5.3 Risoluzione (b)	66
4.5.4 Risoluzione (c)	68
4.5.5 Risoluzione (d)	69

5 Es. 5 - FFDA	70
5.1 Traccia	70
5.2 Strumenti utilizzati	71
5.3 Introduzione - FFDA	71
5.4 Mercury	71
5.4.1 Log Analysis	72
5.4.2 Data manipulation	73
5.4.3 Data Analysis	79
5.4.4 Analisi per Sottosistema	83
5.4.5 Analisi per Layer	90
5.4.6 Analisi dei nodi critici	93
5.5 BlueGene/L	95
5.5.1 Log Analysis	95
5.5.2 Data manipulation	96
5.5.3 Data Analysis	104
5.5.4 Analisi per Rack	108
5.5.5 Analisi per nodi	113
5.5.6 Analisi per Card	116
5.5.7 Analisi del <i>node daemon</i> CIOD	118
5.6 Confronto Mercury - BlueGene/L	120

Es. 1 - Benchmarking

1.1 Traccia

Confrontare le prestazioni di due differenti sistemi (CPUs) sfruttando la libreria di benchmark *LINPACK* per le seguenti dimensioni di N: 1000 5000 10000 20000 30000.

1.2 Strumenti utilizzati

Per la risoluzione di tale esercizio si è fatto uso del software JMP.

1.3 Introduzione

Linpack risolve un sistema di N equazioni lineari simultaneamente, il che è equivalente a risolvere un'equazione vettoriale $A \cdot x = b$, dove x e b sono vettori N-dimensionalì ed A è una matrice $N \times N$.

Nel contesto dell'esercizio, sono stati scelti due sistemi con i seguenti processori:

- Intel i5-8256U @1.60 GHz 4-core 8-thread
- Intel i7-3820QM @2.70 GHz 4-core 8-thread

I due sistemi sono stati confrontati utilizzando come metrica i *gigaflop*: un gigaflop equivale a un miliardo di operazioni in virgola mobile al secondo. Essi vengono generalmente utilizzati per comprendere la velocità del processore e come i computer possono gestire operazioni ad alta intensità di dati che sarebbero comuni in alcuni tipi di processi scientifici o quantitativi.

1.4 Analisi del problema

L'analisi del problema è partita con la creazione di un **precampione** contenente 5 misurazioni indipendenti per ogni valore di cardinalità N scelto. Per garantire che le osservazioni fossero **IID**(indipendenti e identicamente distribuiti), il sistema è stato riavviato prima di ogni esecuzione. In questo modo sono verificate le ipotesi del **teorema del limite centrale**, il quale asserisce che al tendere ad infinito della dimensione di un campione, la

media campionaria tende ad una distribuzione normale. Di conseguenza, essendo valida l'ipotesi di normalità è possibile applicare test parametrici.

Il precampione è stato necessario per poter calcolare la dimensione campionaria per ogni valore di N, fissando il valore di confidenza al 95% e l'errore pari a 5.

Una volta calcolate le misurazioni mancanti in modo da raggiungere la dimensione campionaria precedentemente calcolata, si è proceduto ad effettuare un T-Test paired per ogni valore di N, per poi analizzare la significatività statistica dei campioni differenza. E' stato possibile effettuare un test parametrico come il **T-Test paired** poiché, avendo riavviato i sistemi ogni volta prima di eseguire Linpack, possiamo essere certi dell'indipendenza delle varie osservazioni.

1.5 Soluzione

1.5.1 Stima della dimensione campionaria

La dimensione campionaria n è stata calcolata a partire dal precampione utilizzando la seguente formula:

$$n = \left(\frac{t_{\frac{\alpha}{2}, 4} \cdot \sigma}{E} \right)^2$$

Dove:

- E è l'upper bound dell'errore in gigaflops, tramite il quale effettiamo la stima della dimensione campionaria. Si è scelto E=5.
- σ è approssimato con la deviazione standard del precampione.
- $t_{\frac{\alpha}{2}, 4}$ è il quantile calcolato dalla distribuzione T-student, impostando $\alpha = 0.05$ e 4 gradi di libertà; in questo modo, il quantile è stato calcolato rispetto alla probabilità di coda inferiore $q = 0.975$ e quindi con una confidenza del 95%.

Di seguito sono riportate le dimensioni campionarie calcolate nei due sistemi. Dovendo effettuare un'analisi statistica tramite il T-Test paired, si è scelto per ogni valore di N il massimo valore di dimensione campionaria tra i due sistemi, in modo da avere una stima significativa dei gigaflops per entrambi i processori.

Size	Dim campionaria i7	Dim campionaria i5	Dim campionaria totale
0	1000	10	10
1	5000	6	10
2	10000	10	10
3	20000	8	11
4	30000	5	5

Figura 1.1: Dimensione campionaria sistemi i5 - i7

1.5.2 Generazione campioni differenza

Di seguito sono riportati i campioni differenza al variare della dimensionalità N, calcolati sottraendo coppie di osservazioni per i due sistemi scelti:

Size	GFlops_i7	GFlops_i5	Differenza
0	1000	51.2915	63.0098
1	1000	59.6486	67.7028
2	1000	62.5442	68.8814
3	1000	62.5442	70.3741
4	1000	65.5132	59.7557
5	1000	68.8099	76.7154
6	1000	67.0504	77.9268
7	1000	65.3129	76.1796
8	1000	63.7572	56.1683
9	1000	65.1233	74.7899

Figura 1.2: Campione differenza Size=1000

Size	GFlops_i7	GFlops_i5	Differenza
10	5000	85.4132	90.4942
11	5000	84.9527	93.9367
12	5000	84.6551	84.1415
13	5000	85.1173	97.1382
14	5000	77.2406	85.3702
15	5000	84.6740	94.7838
16	5000	84.6122	94.9458
17	5000	75.8807	88.3601
18	5000	77.7682	86.5526
19	5000	84.6594	92.9364

Figura 1.3: Campione differenza Size=5000

	Size	GFlops_i7	GFlops_i5	Differenza
20	10000	85.4488	52.5843	32.8645
21	10000	86.4364	52.2250	34.2114
22	10000	87.6418	55.8905	31.7513
23	10000	88.3332	46.1603	42.1729
24	10000	88.4634	50.2043	38.2591
25	10000	88.4958	50.0795	38.4163
26	10000	74.8729	44.7120	30.1609
27	10000	88.7145	54.1177	34.5968
28	10000	88.4078	44.7691	43.6387
29	10000	87.5690	45.3876	42.1814

Figura 1.4: Campione differenza Size=10000

	Size	GFlops_i7	GFlops_i5	Differenza
30	20000	73.9988	50.6391	23.3597
31	20000	75.5579	58.9787	16.5792
32	20000	71.7546	43.6755	28.0791
33	20000	85.4732	46.4697	39.0035
34	20000	73.0337	50.3618	22.6719
35	20000	70.9675	60.8296	10.1379
36	20000	78.8415	49.6355	29.2060
37	20000	82.8098	48.5040	34.3058
38	20000	80.3923	47.1058	33.2865
39	20000	73.2465	47.1534	26.0931
40	20000	81.6781	52.1106	29.5675

Figura 1.5: Campione differenza Size=20000

	Size	GFlops_i7	GFlops_i5	Differenza
41	30000	86.8769	11.8920	74.9849
42	30000	86.9886	10.3627	76.6259
43	30000	84.4905	10.2210	74.2695
44	30000	87.9024	11.0516	76.8508
45	30000	85.3501	17.3331	68.0170

Figura 1.6: Campione differenza Size=30000

1.5.3 Analisi statistica in JMP

Tramite l'utilizzo di JMP, si è analizzata la significatività statistica dei vari campioni differenza raccolti. Nel dettaglio, l'obiettivo è quello di dimostrare che le differenze di prestazioni dei due sistemi sono statisticamente significative e che non sono dovute ad effetti aleatori. Per fare ciò, come già detto in precedenza, si è scelto di eseguire un **T-Test paired** con lo scopo di rigettare l'ipotesi nulla (media $\mu = 0$), di fatto uno Zero-Mean Test per ogni campione differenza.

Di seguito sono riportati i risultati del T-Test per i vari campioni differenza.

Campione differenza Size=1000 N=10

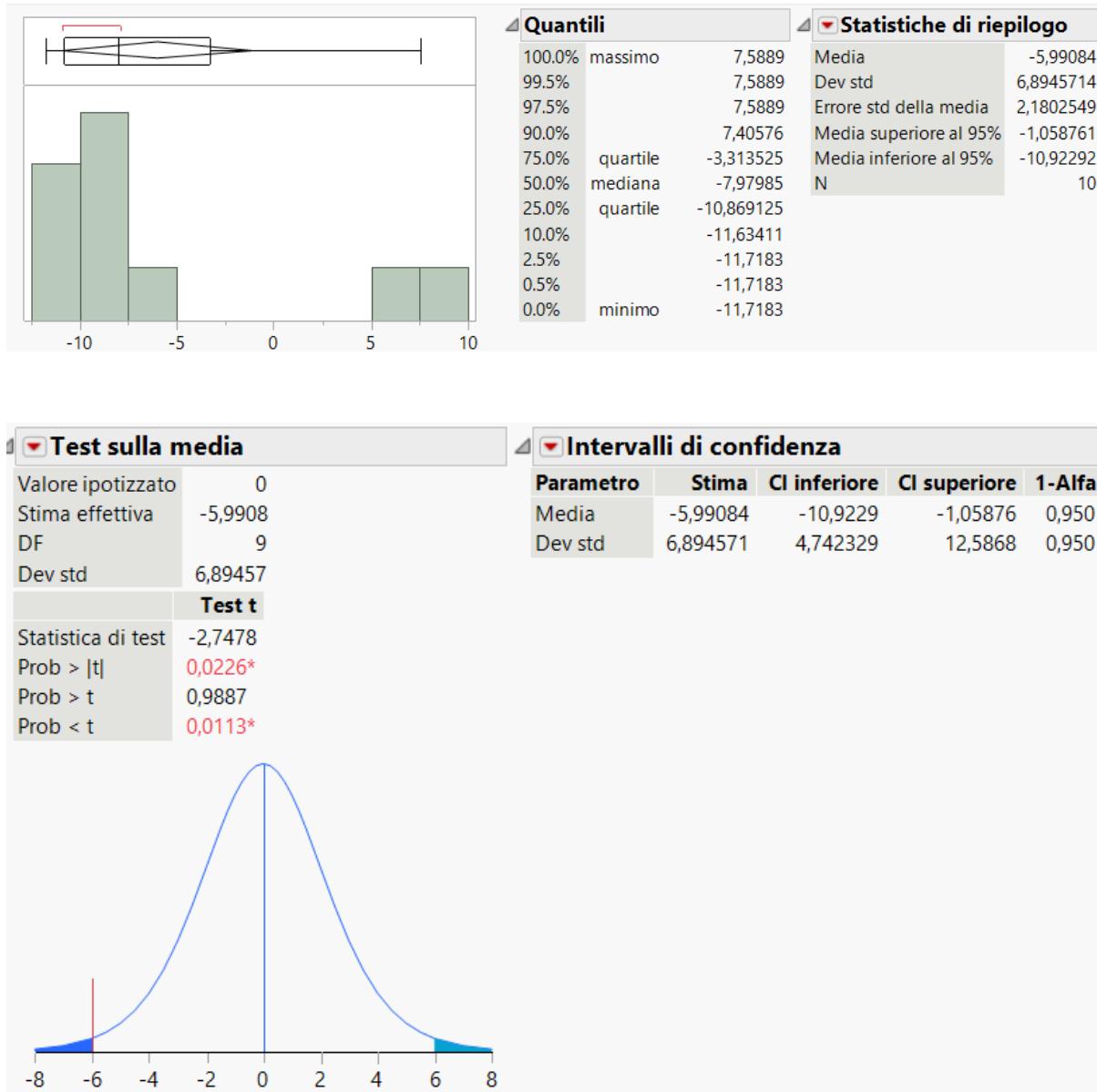


Figura 1.7: Analisi statistica N=1000

Campione differenza Size=5000 N=10

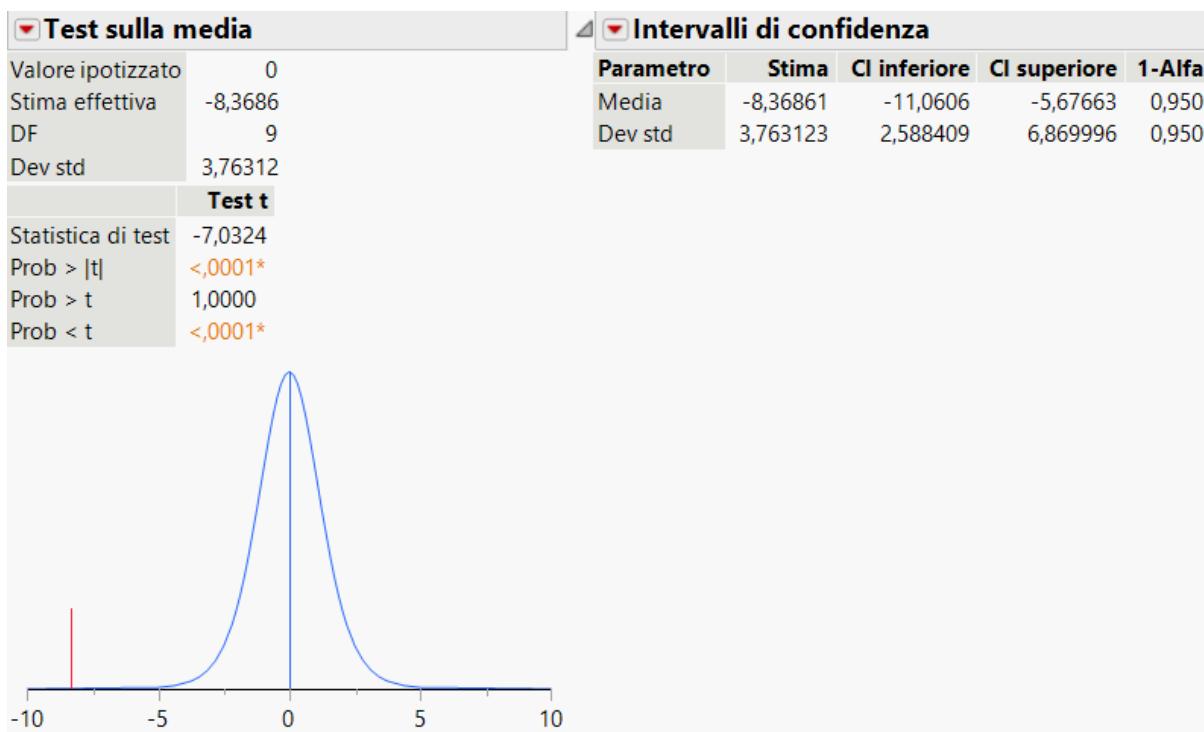
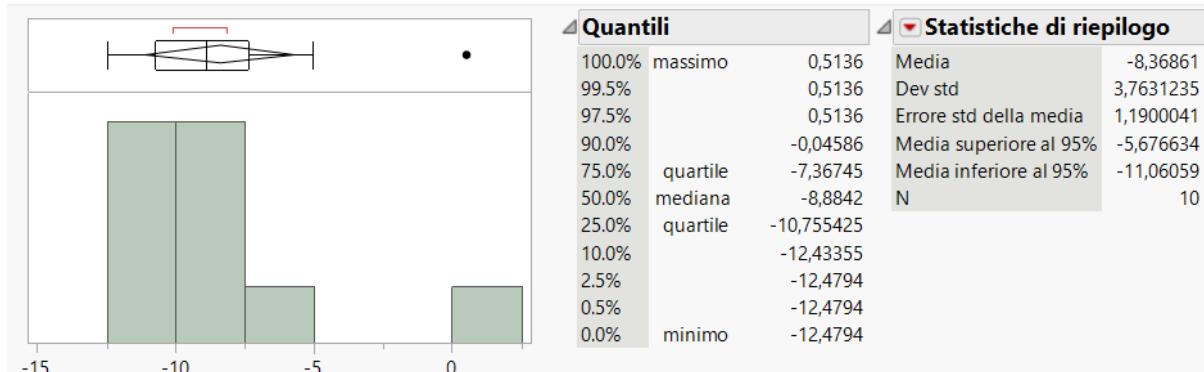


Figura 1.8: Analisi statistica N=5000

Campione differenza Size=10000 N=10

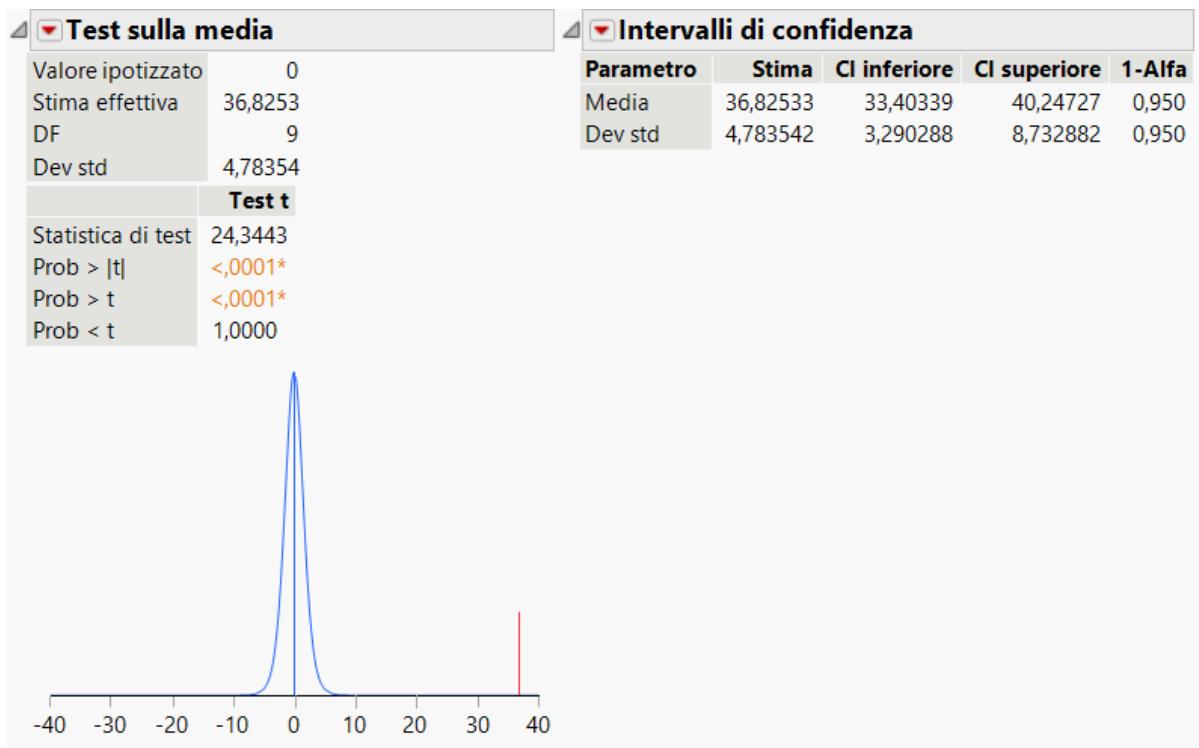
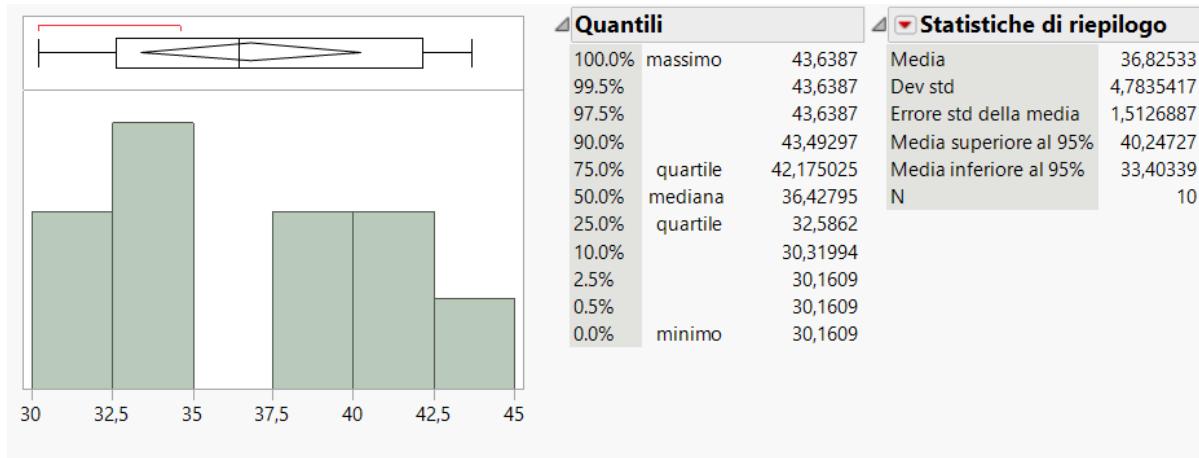


Figura 1.9: Analisi statistica N=10000

Campione differenza Size=20000 N=11

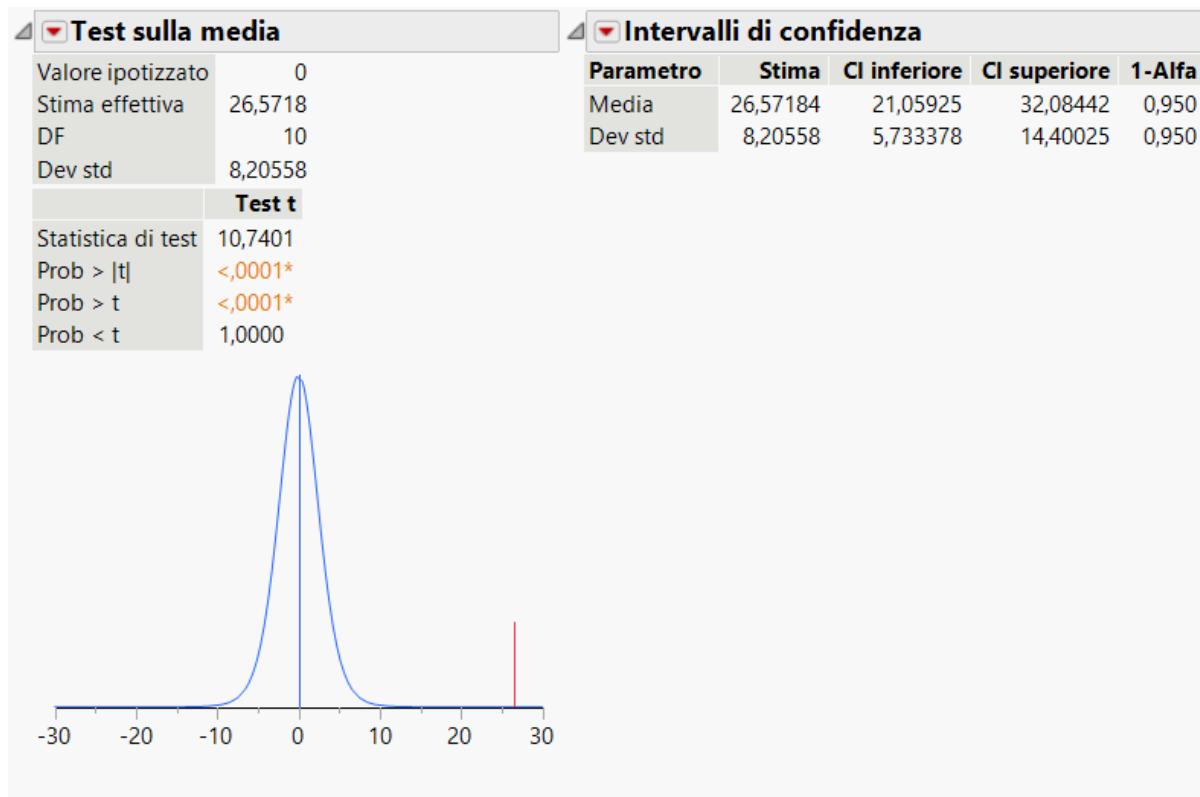
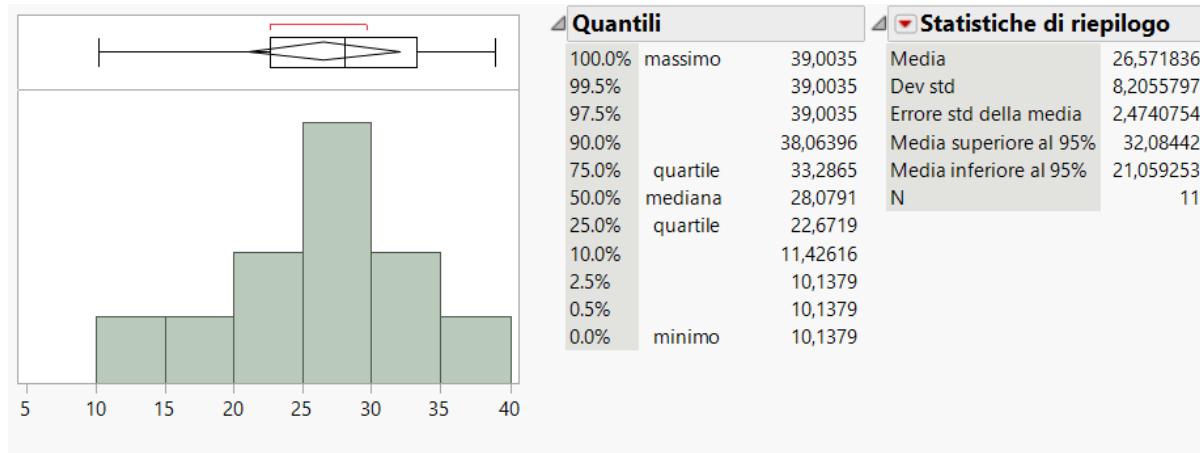


Figura 1.10: Analisi statistica N=20000

Campione differenza Size=30000 N=5

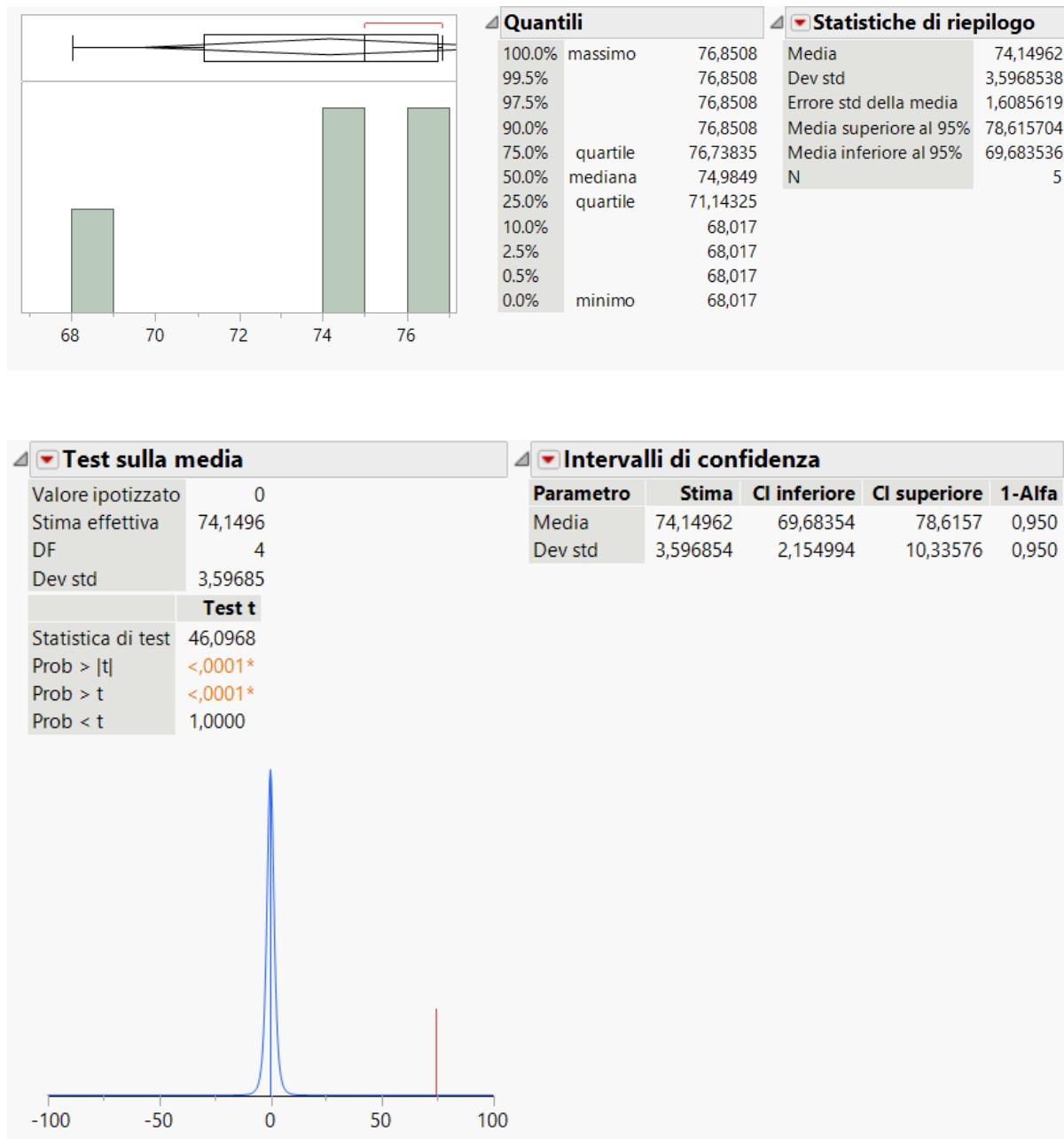


Figura 1.11: Analisi statistica N=30000

1.5.4 Considerazioni

Indipendentemente dal campione differenza analizzato, è possibile concludere che:

- dall'istogramma frequentistico si può notare come le osservazioni del campione differenza non seguono una distribuzione normale. Tuttavia ciò non è di interesse poiché la distribuzione normale è da ricercare per valori di dimensione campionaria almeno superiori a 30, mentre nel nostro caso il teorema del limite centrale è comunque valido avendo garantito l'indipendenza delle osservazioni.
- Si può inoltre confermare la validità del teorema, e quindi l'indipendenza dei campioni, utilizzando come indicatore la similarità tra i valori di media e mediana. Questo indicatore trova corrispondenza in tutti i test effettuati.
- Effettuando lo **Zero-Mean Test** in cui è specificata l'ipotesi nulla $H_0 : \mu = 0$ e un valore di soglia $\alpha = 0.05$, si può procedere all'analisi dei risultati restituiti da JMP, in particolare del *p-value* riportato sotto la voce Prob $\leq t$. Il p-value è definito in statistica inferenziale come la probabilità di ottenere risultati uguali o meno probabili di quelli osservati durante il test, supposta vera l'ipotesi nulla. Quindi il valore p aiuta a capire se la differenza tra il risultato osservato e quello ipotizzato è dovuta alla casualità introdotta dal campionamento, oppure se tale differenza è statisticamente significativa. Nel dettaglio, se risulta che:

$$p - value < \alpha$$

l'evidenza empirica è fortemente contraria all'ipotesi nulla, la quale viene rigettata. Si noti come in tutti i test effettuati per i diversi campioni differenza portano al rigetto dell'ipotesi nulla, dunque si può concludere che esiste realmente una differenza statisticamente significativa tra le prestazioni dei due sistemi presi in esame, con una confidenza del 95% e un errore di 5 gigaflops.

- un ulteriore conferma dell'analisi svolta può essere ottenuta tramite l'utilizzo degli intervalli di confidenza: notiamo come negli intervalli di ciascun test non è mai compreso lo 0, quindi si è confidenti al 95% con un errore di 5 gigaflops che non sussiste l'ipotesi nulla e quindi c'è una differenza statistica dovuta a un fenomeno e non a una casualità.

Es. 2 - Workload Characterization

2.1 Traccia

A partire dai dati del workload reale, trovare un workload sintetico (caratterizzato) con le seguenti proprietà:

- trovare il miglior trade-off tra devianza persa (*deviance loss*) e riduzione della dimensionalità (*data size reduction*)

2.2 Strumenti utilizzati

L'analisi di tale esercizio è stata affrontata utilizzando notebook *Python* sviluppati nell'ambiente Jupyter Notebook.

2.3 Introduzione

In questo esercizio di Workload Characterization, l'obiettivo è per l'appunto quello di caratterizzare un workload, denominato workload reale, andando a costruire un dataset ridotto, detto workload sintetico, che sia rappresentativo di quello di partenza.

Il processo di caratterizzazione mira a estrarre un *modello statistico* dal workload reale, ossia un modello che ne rappresenti tutta la variabilità esistente nei dati, cercando allo stesso tempo di filtrare qualsiasi componente aleatoria insita in essi.

Le tecniche che sono state utilizzate per raggiungere tale scopo, sono quelle della PCA (Principal Component Analysis) e del Clustering.

L'analisi delle componenti principali (**PCA**) è un metodo rientrante nei problemi di trasformazione lineare che viene ampiamente utilizzata in diversi campi, tra cui quello per la riduzione della dimensionalità. La PCA permette di trovare le direzioni della massima varianza nei dati ad alta dimensione e di proiettarle su un nuovo sottospazio con dimensioni uguali o inferiori a quello originale, in cui le nuove variabili in gioco sono denominate *componenti principali*.

Il **Clustering** è invece un processo per organizzare un insieme di "oggetti" in gruppi, detti cluster, i cui membri sono in qualche modo simili tra loro e dissimili rispetto ai componenti degli altri cluster. Tale tecnica è stata utilizzata per ridurre la dimensione del dataset reale, tramite la suddivisione di quest'ultimo in cluster e la scelta di campioni rappresentativi per ognuno di essi. Nel dettaglio, ci si è soffermati esclusivamente sulle

tecniche di clustering di tipo gerarchico, in quanto ci permettono di avere un fattore di scelta in più per la realizzazione del trade-off tra devianza persa e dimensionalità.

2.4 Analisi del problema

L'analisi del problema è partita dalla valutazione del workload reale, composto da 3000 righe e 24 colonne.

	VmPeak	VmSize	VmHWM	VmRSS	VmPTE	...	proc-fd	avgThroughput	avgElapsed	avgLatency	Errors
0	144012	126424	14224	14224	108	...	6968	268820	510	0	2
1	152272	150216	24508	23708	160	...	7192	293700	510	0	2
2	152272	150216	24628	24588	160	...	7188	293336	510	0	2
3	152272	150216	24628	24500	160	...	7196	293336	510	0	2
4	152272	150216	24628	24580	160	...	7196	295484	1020	0	2

Figura 2.1: Estratto dal workload reale (3000×24)

In primo luogo, per la riduzione della dimensionalità (numero di feature) è stata studiata la correlazione tra le colonne del dataset iniziale, in modo da eliminare le colonne perfettamente correlate e quelle costanti per applicare l'analisi delle componenti principali. A valle della PCA, si è proceduto con la clusterizzazione del workload reale, utilizzando il metodo di Ward. Per concludere, è stata fatta un'analisi di sensitività dei dati ottenuti per trovare il miglior trade-off tra devianza persa e riduzione della dimensionalità.

2.5 Soluzione

2.5.1 Studio della correlazione

Per studiare la correlazione tra le varie colonne del workload reale si è deciso di fare un filtro a monte della PCA calcolando la matrice di correlazione per eliminare le colonne perfettamente correlate (caratterizzate da 1 fuori diagonale) e le colonne costanti. Questa operazione si è resa necessaria per fare in modo che, quando successivamente si andrà a tracciare lo Scree Plot, non siano presenti componenti principali che non spieghino alcuna percentuale di varianza.

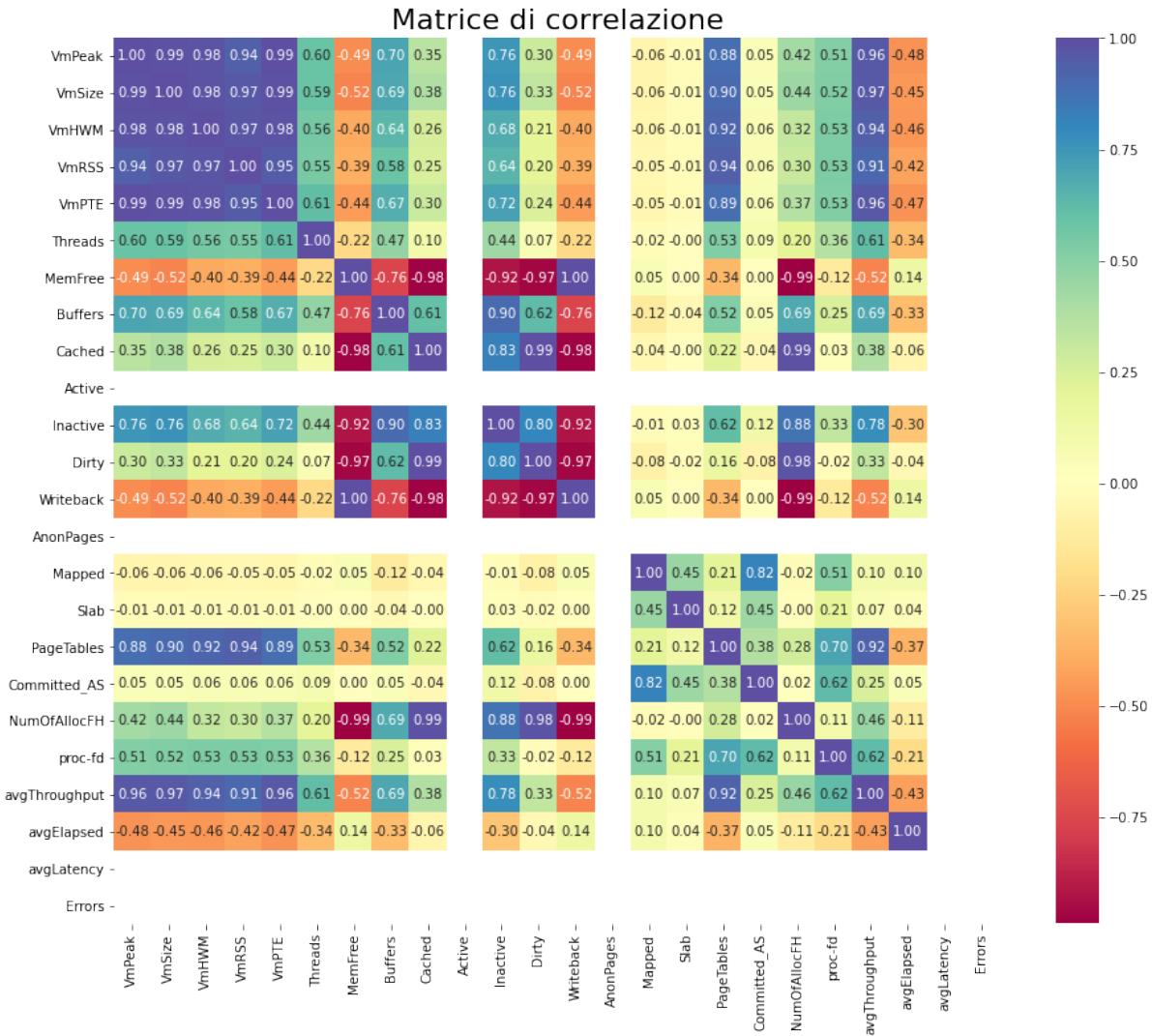


Figura 2.2: Matrice di correlazione

Le colonne *Active*, *AnonPages*, *avgLatency*, *Errors* sono costanti, come emerge dalla matrice di correlazione. Quindi possono essere rimosse senza perdere informazioni.

Le colonne *MemFree* e *WriteBack* sono invece perfettamente correlate, come si nota dalla matrice di correlazione che riporta un 1 fuori diagonale principale. Quindi si è scelto di rimuovere *MemFree* in quanto l'informazione in essa contenuta è ridondante.

2.5.2 Principal Component Analysis

In prima battuta, si è reso necessario effettuare una **standardizzazione Z-score** del workload, in modo che i nuovi dati ottenuti siano caratterizzati da media nulla e varianza unitaria.

$$x'_i = \frac{x_i - \mu_X}{\sigma_X}$$

Tale normalizzazione è importante per poter effettuare delle valutazioni statistiche che non siano polarizzate dai valori assoluti dei dati in gioco.

A questo punto è stato possibile utilizzare la PCA sui dati appena modificati in modo da ottenere il cosiddetto Scree Plot, che mostra la percentuale di varianza spiegata da ogni componente principale PC_i rispetto alla varianza totale del dataset iniziale.

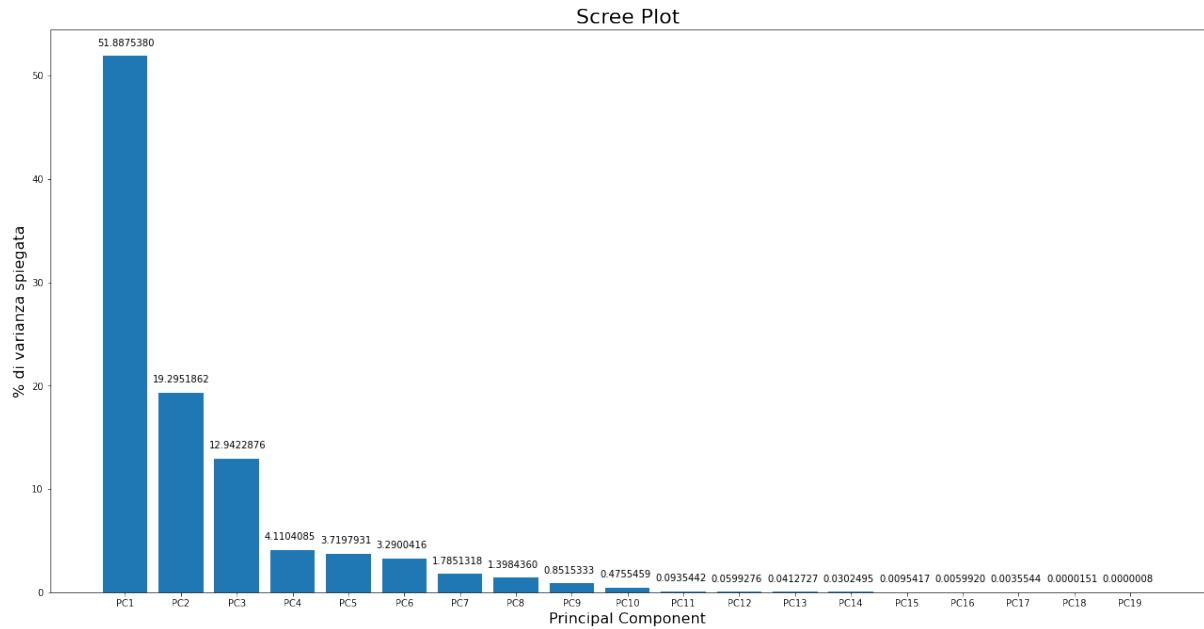


Figura 2.3: Scree Plot

Per ottenere la percentuale di varianza spiegata per ogni componente principale si frutta la seguente formula:

$$\%var_i = \frac{\lambda_i}{\sum_{i=1}^{19} \lambda_i}$$

Dove i λ_i sono gli autovalori della matrice di correlazione, preventivamente ordinati in senso decrescente:

$$\begin{aligned}
 \lambda_1 &= 9.861919529674706 & \lambda_2 &= 3.6673078161074426 \\
 \lambda_3 &= 2.4598545970856605 & \lambda_4 &= 0.7812380276339967 \\
 \lambda_5 &= 0.7069963579187262 & \lambda_6 &= 0.6253163492856347 \\
 \lambda_7 &= 0.33928813324565044 & \lambda_8 &= 0.2657914279935232 \\
 \lambda_9 &= 0.16184527157118334 & \lambda_{10} &= 0.09038383972819888 \\
 \lambda_{11} &= 0.017779323793995157 & \lambda_{12} &= 0.011390035733941876 \\
 \lambda_{13} &= 0.007844429474656163 & \lambda_{14} &= 0.005749317890895283 \\
 \lambda_{15} &= 0.0018135314022888385 & \lambda_{16} &= 0.0011388684300241605 \\
 \lambda_{17} &= 0.0006755696099951507 & \lambda_{18} &= 2.8757321215386666 \cdot 10^{-6} \\
 \lambda_{19} &= 1.4283572519182327 \cdot 10^{-7}
 \end{aligned}$$

Da tali dati ne consegue la seguente tabella, in cui è riportata la percentuale di varianza spiegata rispetto al numero di componenti principali:

# PC	% varianza
1	51.887538
2	71.182724
3	84.125012
4	88.235420
5	91.955213
6	95.245255
7	97.030387
8	98.428823
9	99.280356
10	99.755902
11	99.849446
12	99.909374
13	99.950646
14	99.980896
15	99.990438
16	99.996430
17	99.999984
18	99.999999
19	100.000000

Tabella 2.1: Percentuale di varianza spiegata

Uno strumento grafico per supportare la scelta delle componenti principali da considerare è il **loading plot**, il quale mostra come sono rappresentate le componenti principali del workload nel nuovo sistema di riferimento, basato su $PC1$ e $PC2$.

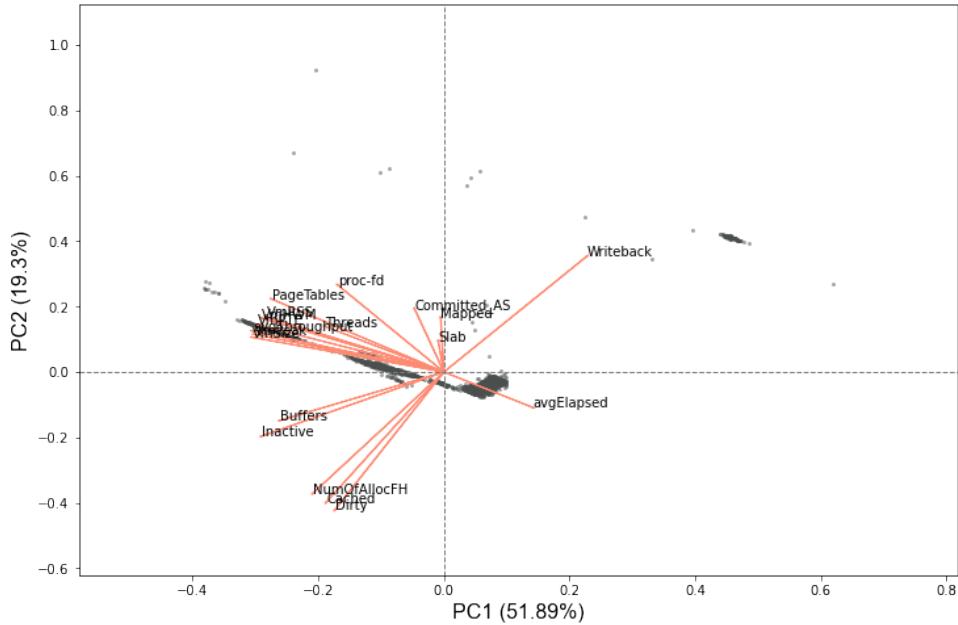


Figura 2.4: Loading Plot

Da questo grafico è possibile effettuare un primo controllo visivo del problema, notando che ci sono alcune componenti molto correlate (vicine tra loro): una prima intuizione visiva potrebbe suggerire la presenza circa 4-6 componenti principali che contribuiscono alla varianza (1 nel I quadrante, 2-3 nel II, 1-2 nel III e 1 nel IV).

2.5.3 Clustering

A valle della PCA si procede con la clusterizzazione che, come già detto, si rende necessaria per realizzare la riduzione dei punti del dataset. Ricordiamo che tale tecnica permette la suddivisione del dataset di partenza in sottogruppi di oggetti, denominati cluster, i quali sono simili tra loro e dissimili dai membri degli altri cluster. Tuttavia è necessario definire in qualche modo il concetto di similarità (e dissimilarità) tra gli elementi del workload, e tale definizione è chiamata metrica di distanza. Esistono differenti metriche di distanza in letteratura, tuttavia nel metodo di clustering scelto, ossia il **metodo di Ward**, la metrica adottata è la somma dei quadrati (SS, nient'altro che la distanza euclidea al quadrato) che è proprio la devianza, e quindi si ottiene una clusterizzazione che minimizza la devianza dei punti di uno stesso cluster: l'obiettivo è quindi accorpare in un cluster i diversi punti a devianza minima, per minimizzare la devianza intra-cluster e massimizzare la devianza inter-cluster. In tal modo, essendo i punti di un cluster a devianza minima, scegliendone uno di essi, tale punto sarà abbastanza rappresentativo del proprio cluster di appartenenza. Si noti come è vantaggioso studiare la devianza in quanto rispetta la disuguaglianza triangolare, a differenza della varianza, da cui risulta la seguente proprietà:

$$D_{TOT} = D_{INTRA} + D_{INTER}$$

Tuttavia ciò non rappresenta un grosso problema in quanto minimizzando la devianza intra-cluster, si minimizza di conseguenza anche la varianza intra-cluster; tale ragionamento non può però essere esteso alla devianza inter-cluster.

Il metodo di Ward, inoltre, è un metodo gerarchico e agglomerativo, ed il risultato può essere graficato come un dendrogramma, ossia un albero binario in cui la radice rappresenta l'intero dataset come un unico cluster, le foglie rappresentano i singoli elementi del dataset, ognuno in un cluster a sé stante, mentre i nodi intermedi rappresentano i cluster determinati ai vari livelli di altezza, ossia a differenti livelli di distanza minima.

Applicando il metodo di Ward al workload reale trasformato dalla PCA si ottiene dunque il seguente dendrogramma:

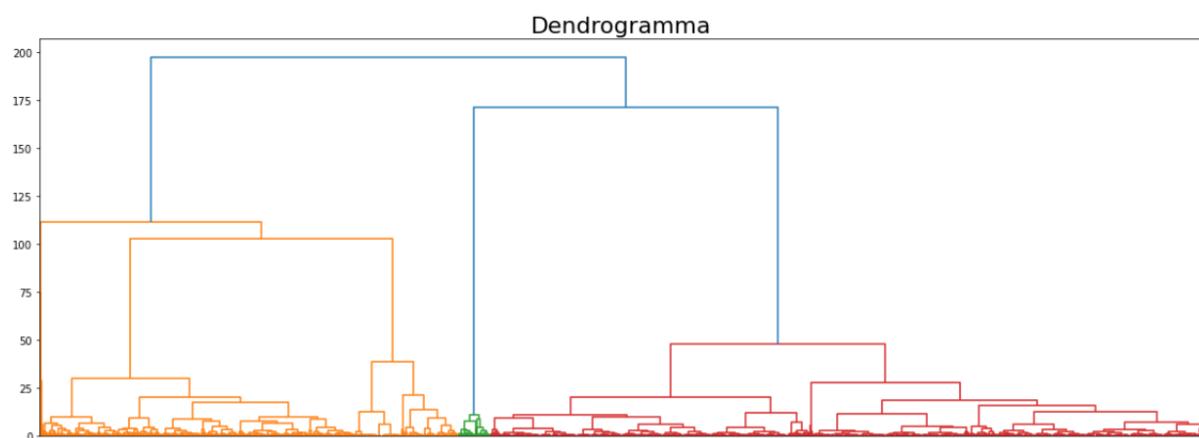


Figura 2.5: Dendrogramma

E troncandolo possiamo ottenere una visualizzazione migliore, nella quale si mette in evidenza anche la distanza a cui vengono accorpati i cluster:

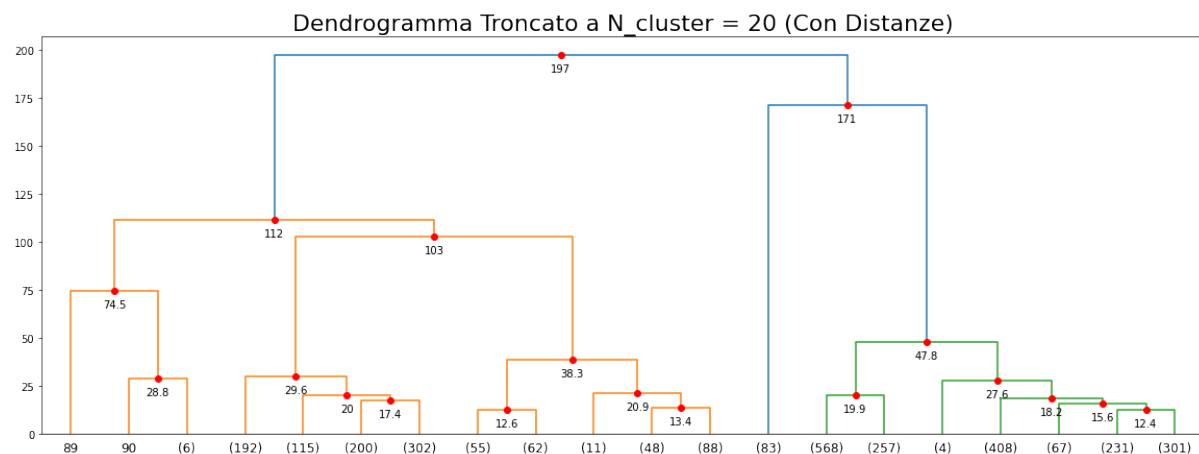


Figura 2.6: Dendrogramma troncato a 20 cluster

# Cluster	Distanza
1	197.209308
2	171.339476
3	111.589392
4	102.785601
5	74.464585
6	47.764839
7	38.261090
8	29.634771
9	28.819799
10	27.594849
11	20.853332
12	19.961865
13	19.855290
14	18.228172
15	17.374125
16	15.647187
17	13.357968
18	12.554688
19	12.358143

Tabella 2.2: Distanza accorpamento cluster

2.5.4 Valutazione del Trade-off

Ultimate le analisi tramite le tecniche di PCA e clustering, si è proceduto ad effettuare un trade-off tra la riduzione della dimensione del dataset e la devianza intra-cluster persa, al variare del numero di cluster e del numero delle componenti principali.

In prima battuta, si è ritenuto opportuno tenere in considerazione un numero di componenti principali compreso tra 3 e 6, in modo tale da spiegare una percentuale di varianza totale, a valle della PCA, compresa nella fascia 84%-95%. Per quanto riguarda il numero il cluster, si è analizzato un intervallo abbastanza generico, compreso tra 1 e 20.

Per agevolare la scelta, è stato innanzitutto realizzato il grafico rappresentante una famiglia di curve(una per ogni gruppo di PC scelte) della devianza persa in relazione al numero di cluster.

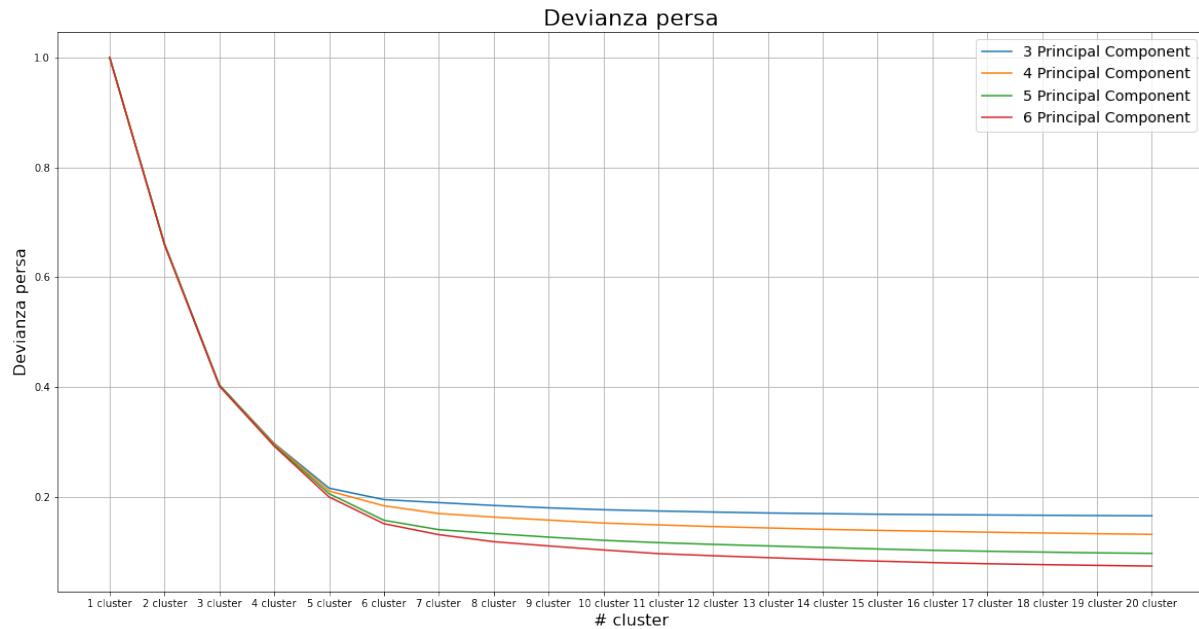


Figura 2.7: Grafico devianza persa

Successivamente, l'analisi è stata integrata con il grafico in cui viene presa in considerazione, oltre all'informazione contenuta nel grafico precedente, anche il *dimension ratio*, ossia il rapporto tra la dimensione del workload sintetico rispetto al workload reale. Per permettere una valutazione oggettiva e non polarizzata da tale parametro, si è proceduto a normalizzare il dimension ratio sul rispettivo valore massimo. In tal modo, sia la devianza persa sia il dimension ratio sono entrambi scalati nell'intervallo [0, 1] ed hanno lo stesso peso ai fini dell'analisi.

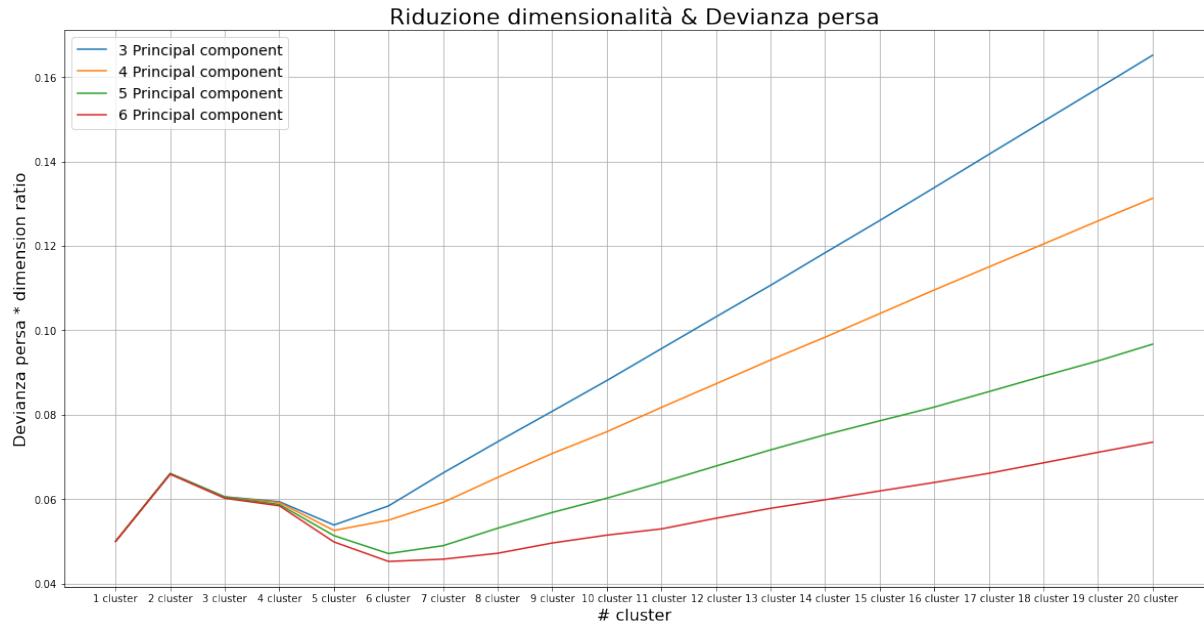


Figura 2.8: Grafico devianza persa & riduzione di dimensionalità

Come si evince da tale grafico, il miglior trade-off tra la dimensionalità del workload sintetico ($\# \text{ PC} \times \# \text{ cluster}$) e la devianza persa è rappresentato dalla scelta di 6 Componenti Principali e di 6 cluster, in quanto questa configurazione minimizza la devianza persa e massimizza la riduzione del workload.

2.5.5 Workload sintetico

Per concludere l'esercizio, il trade-off appena scelto è stato applicato al workload reale, in modo da poter finalmente ottenere il workload sintetico.

È stato quindi realizzato il dendrogramma finale, tagliato all'altezza di 6 cluster:

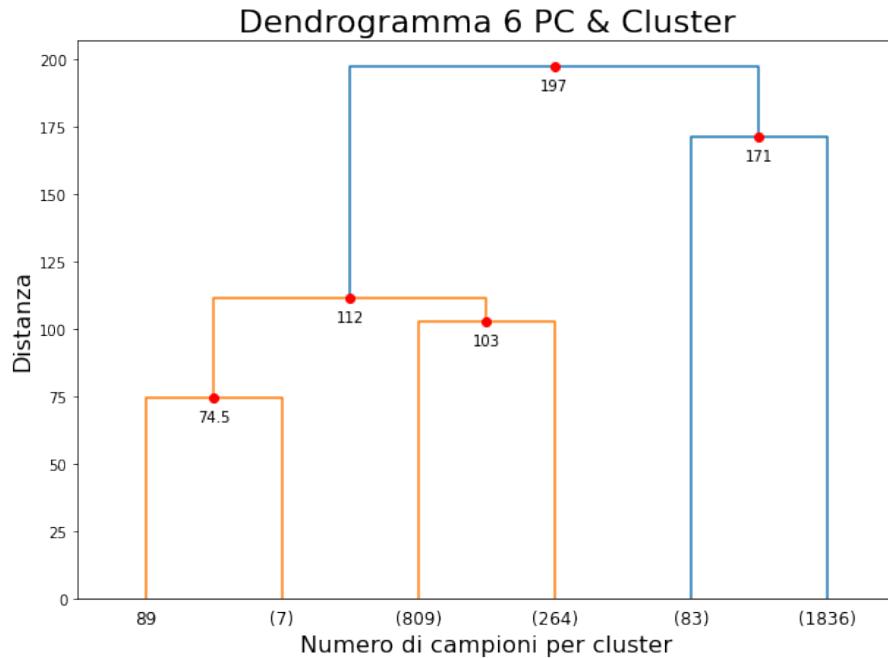


Figura 2.9: Dendrogramma tagliato a 6 cluster

Al fine di ridurre la dimensione dei punti, è stato campionato randomicamente un solo elemento per ogni cluster, ottenendo il seguente dataset ridotto, ancora nel dominio della Principal Component Analysis:

	PC1	PC2	PC3	PC4	PC5	PC6
45	11.028302	8.365460	-2.460798	-0.135171	0.326483	-1.151638
89	-4.891286	18.927369	52.301081	21.015948	28.859324	-3.075886
90	-5.741748	13.756474	37.031094	-4.944510	-14.533539	3.471819
1228	1.227460	-1.022921	0.265429	0.587693	-0.352627	0.492260
2021	-0.945396	-0.389603	0.157341	-0.653055	0.280823	-0.908295

Figura 2.10: Workload sintetico nel dominio della PCA

Con questa configurazione è stato ridotto il dataset iniziale del 99.8% conservando l'84.9% della varianza dei dati iniziali.

L'esercizio è stato quindi concluso ritornando nel dominio delle feature iniziali, recuperando i valori precedenti all'operazione di normalizzazione. Il workload sintetico definitivo è riportato di seguito:

	VmPeak	VmSize	VmHWM	VmRSS	VmPTE	...	proc-fd	avgThroughput	avgElapsed	avgLatency	Errors
45	152272	150216	25228	25208	160	...	7192	294280	1530	0	2
89	170344	170340	31144	31140	200	...	8028	365264	2040	0	2
90	170344	170340	31144	31140	200	...	8024	365692	1530	0	2
1228	172392	170340	32868	30848	200	...	7208	316640	1020	0	2
2021	186012	178504	41320	35864	224	...	7232	327268	1530	0	2

Figura 2.11: Workload sintetico

Es. 3 - Web Server Performance Analysis

3.1 Traccia

Analizzare le performance di un web server, effettuando le seguenti attività:

- *Workload Characterization*: estrapolare un workload sintetico a partire da un workload reale e verificarne la significatività statistica dei rispettivi workload di basso livello.
- *Capacity Test*: valutazione delle performance del sistema al variare del carico di lavoro imposto.
- *Design of Experiment*: verifica dell’infuenza dei fattori sul response time del sistema.

3.2 Strumenti utilizzati

Per la realizzazione delle analisi di *Workload Characterization* e *Capacity Test* sono stati utilizzati dei notebook *Python*, sviluppati nell’ambiente Jupyter Notebook. Per il *Design of Experiment* si è invece fatto uso del software JMP.

3.3 Workload Characterization

La Workload Characterization nel contesto dell’analisi delle performance di un Web Server consiste in una serie di fasi che permettono di verificare la significatività statistica del workload sintetico, generato a partire dal workload reale.

Nella prima fase, è necessario definire il SUT (System Under Test), che in questo caso è un Web Server Apache situato su una Oracle Virtual Machine, caratterizzata dalle seguenti specifiche:

- Sistema Operativo: *Ubuntu 19.04 64 bit*
- RAM: 1 GB
- CPU: *Intel I5-8256U*

Su tale server sono posizionate 3 risorse di dimensioni differenti, le quali possono essere richieste dal client tramite delle richieste HTTP. Le risorse scelte per tale analisi sono le seguenti:

- Risorsa BIG: *andromeda_big.jpg*, 2.8 MB
- Risorsa MEDIUM: *cassiopea_mid.jpg* 0.9 MB
- Risorsa SMALL: *jupiter_small.jpg* 150 KB

Dal sistema appena definito è necessario collezionare due insiemi di feature a due livelli di dettaglio differente: le feature di alto livello (HL), le quali sono collezionate lato client tramite il software Apache JMeter; le feature di basso livello (LL), raccolte lato server direttamente dal terminale. La fase 1 si conclude con la caratterizzazione dei workload appena raccolti (generando i workload sintetici LL_c e HL_c).

Nella seconda fase, il workload sintetico relativo ai parametri di alto livello è posto in ingresso al sistema per generare un nuovo workload di basso livello (LL'), il quale è stato opportunamente caratterizzato (generando LL'_c).

Nella terza ed ultima fase, vengono confrontati i due workload sintetici di LL generati nelle fasi precedenti, con l'obiettivo di verificare la significatività statistica di LL'_c rispetto a LL_c e permettere la validazione del workload sintetico HL_c .

3.3.1 Fase 1: Workload Characterization HL e LL

Workload Characterization HL

La raccolta dei dati di HL è stata effettuata tramite il software Apache Jmeter, cercando di emulare quanto più possibile un workload reale tramite la somministrazione di differenti richieste al server in esame. Nel dettaglio, sono stati impostati 3 Thread Group, ognuno contenente richieste a tutte le risorse del server, effettuate casualmente tramite l'utilizzo di un Random Controller. I Thread Group sono caratterizzati da configurazioni differenti, riportate qui di seguito:

# Thread Group	Throughput	Threads	Ramp-up
TG1	120	30	30
TG2	60	60	120
TG3	30	90	270

Tabella 3.1

La durata del test è stata impostata a 300 secondi.

Di seguito si riporta un estratto dei dati di alto livello collezionati:

	timeStamp	elapsed	label	responseCode	responseMessage	threadName	dataType	success	failureMessage	bytes	sentBytes	grpThreads	allThreads	URL	Latency	IdleTime	Connect
0	1608199962067	7	HTTP Request small	200	OK	Thread Group 1-1	bin	True	NaN	152918	133	1	1	http://192.168.56.101/jupiter_small.jpg	4	0	2
1	1608199962116	185	HTTP Request small	200	OK	Thread Group 2-1	bin	True	NaN	152918	133	1	3	http://192.168.56.101/jupiter_small.jpg	116	0	115
2	1608199962211	93	HTTP Request small	200	OK	Thread Group 3-1	bin	True	NaN	152918	133	1	3	http://192.168.56.101/jupiter_small.jpg	29	0	20
3	1608199962567	26	HTTP Request mid	200	OK	Thread Group 1-1	bin	True	NaN	1031744	133	1	3	http://192.168.56.101/cassiopea_mid.jpg	2	0	0
4	1608199963070	6	HTTP Request small	200	OK	Thread Group 1-2	bin	True	NaN	152918	133	2	4	http://192.168.56.101/jupiter_small.jpg	3	0	0
...
1222	1608200261212	306	HTTP Request big	200	OK	Thread Group 3-2	bin	True	NaN	2962330	133	2	5	http://192.168.56.101/andromeda_big.jpg	6	0	3
1223	1608200261258	271	HTTP Request big	200	OK	Thread Group 3-3-8	bin	True	NaN	2962330	133	1	4	http://192.168.56.101/andromeda_big.jpg	6	0	2
1224	1608200261567	12	HTTP Request small	200	OK	Thread Group 1-1-1	bin	True	NaN	152918	133	3	3	http://192.168.56.101/jupiter_small.jpg	5	0	3
1225	1608200261568	27	HTTP Request small	200	OK	Thread Group 1-1-3	bin	True	NaN	152918	133	2	2	http://192.168.56.101/jupiter_small.jpg	8	0	4
1226	1608200261579	51	HTTP Request big	200	OK	Thread Group 1-1-4	bin	True	NaN	2962330	133	1	1	http://192.168.56.101/andromeda_big.jpg	2	0	1

Figura 3.1: Estratto workload HL

Jmeter ha collezionato 1227 entry con 17 feature. Per procedere con la caratterizzazione del workload appena ottenuto, si è reso necessario un filtraggio delle feature in modo da preservare esclusivamente le informazioni di interesse ed eliminando i dati categorici. Le feature preservate in seguito al filtraggio sono:

- *timeStamp*
- *elapsed*, tempo che intercorre tra l'invio della richiesta e il completamento della risposta
- *label*, contenente informazioni sulla tipologia di richiesta effettuata
- *threadName*, che indica il Thread Group con lo specifico thread che ha effettuato la richiesta
- *bytes*, dimensione della richiesta in byte
- *Latency*, tempo che intercorre tra l'invio della richiesta e l'inizio della risposta

Le feature *label* e *threadName* sono state unite in modo da riassumere in un'unica colonna le informazioni riguardanti la tipologia di richiesta (small, medium, big) e il Thread Group che l'ha effettuata (1,2,3). Tali informazioni sono state poi codificate in modo da permettere l'analisi delle componenti principali:

#	label-threadName	Codifica
	big-1	0
	big-2	1
	big-3	2
	mid-1	3
	mid-2	4
	mid-3	5
	small-1	6
	small-2	7
	small-3	8

Si è quindi ottenuto il seguente workload HL filtrato:

	timeStamp	elapsed	bytes	Latency	size_group
0	1608199962067	7	152918	4	6
1	1608199962116	185	152918	116	7
2	1608199962211	93	152918	29	8
3	1608199962367	26	1031744	2	3
4	1608199963070	6	152918	3	6
...
1222	1608200261212	306	2962330	6	2
1223	1608200261258	271	2962330	6	2
1224	1608200261567	12	152918	5	6
1225	1608200261568	27	152918	8	6
1226	1608200261579	51	2962330	2	0

Figura 3.2: Workload HL filtrato

L'analisi a componenti principali restituisce le seguenti percentuali di varianza spiegata, riportate nello Scree Plot di seguito. Dato che le componenti principali a disposizione sono soltanto 5, si è scelto di scegliere le prime 3 PC, le quali sono in grado di spiegare l'89.4% della varianza totale dei dati.

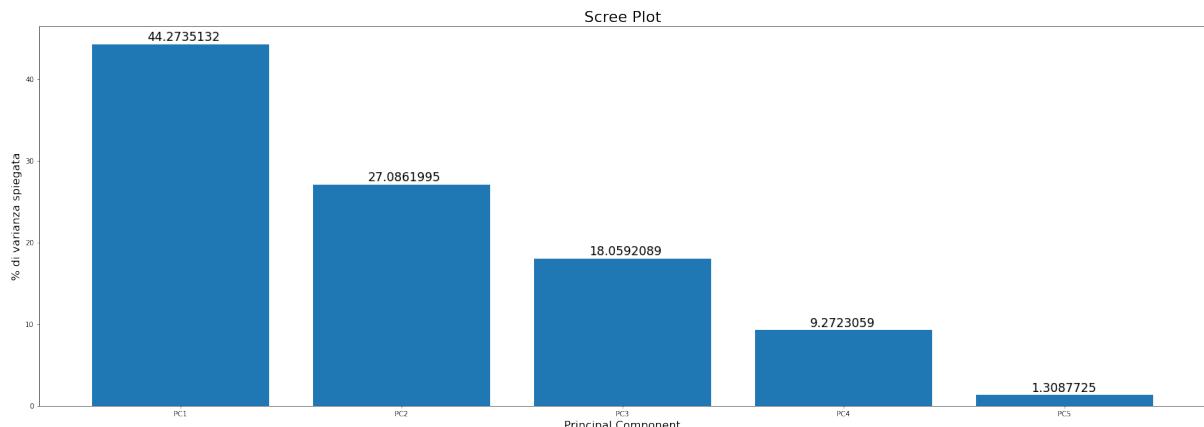


Figura 3.3: PCA - Scree Plot

La successiva fase di clustering è stata effettuata analizzando il grafico della devianza persa in relazione al numero di cluster scelti:

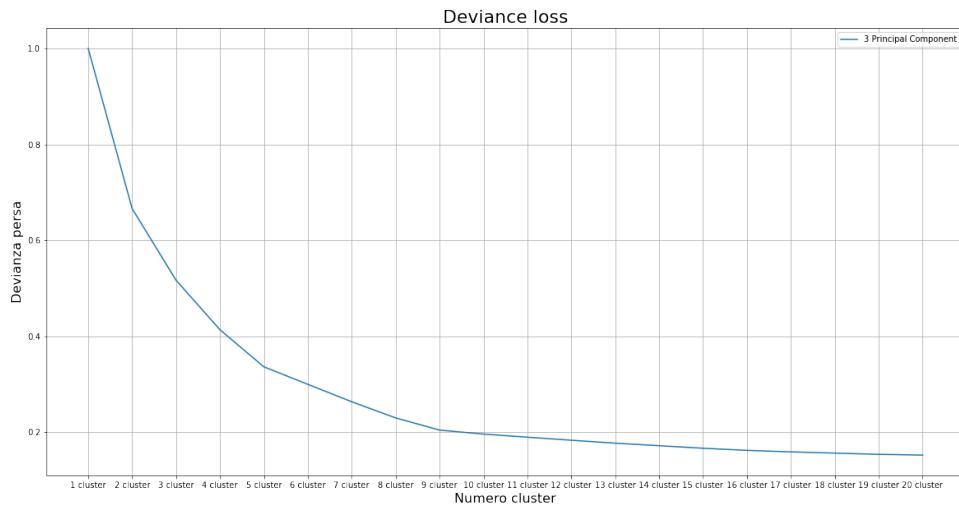


Figura 3.4: Clustering - Devianza persa

Il gomito della curva si trova in corrispondenza di 5 cluster, il che ci permette di concludere le fasi di PCA e Clustering con una percentuale di devianza spiegata totale pari al 66.4%.

A questo punto, per ultimare il processo di Workload Characterization si è proceduto a campionare per ogni cluster uno specifico esperimento. Tuttavia, anziché selezionare casualmente tale punto all'interno dei vari cluster, si è preferito scegliere per ogni cluster l'esperimento più rappresentativo del cluster stesso, scegliendo tra le configurazioni *label-threadName* più frequenti in esso.

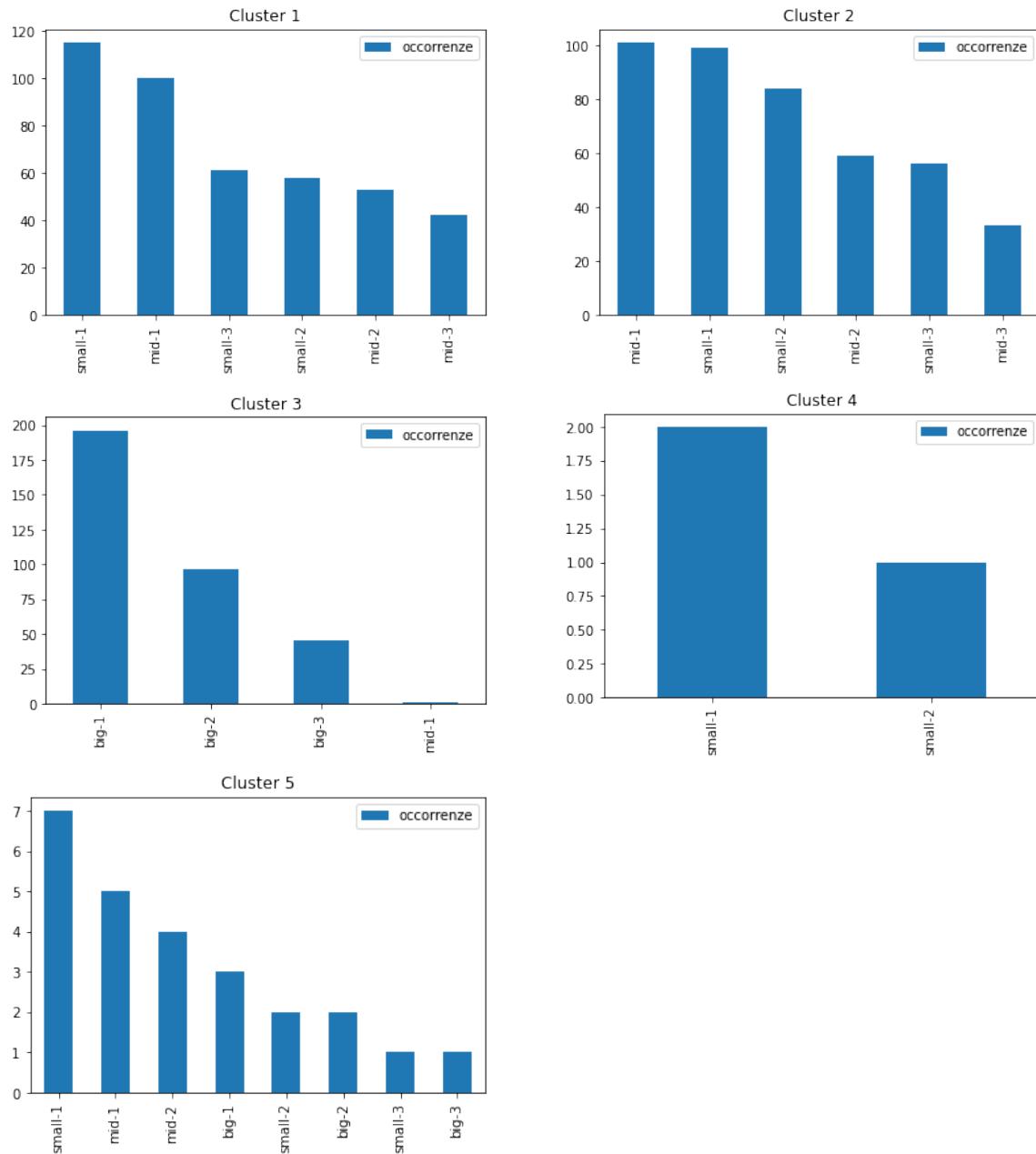


Figura 3.5: Frequenza configurazioni *label-threadName* per cluster

Tali considerazioni hanno portato alla scelta delle seguenti configurazioni per ogni cluster:

- Cluster 1: configurazione *small-1*
- Cluster 2: configurazione *mid-1*
- Cluster 3: configurazione *big-1*
- Cluster 4: configurazione *small-2*
- Cluster 5: configurazione *small-1*

Per ognuna di tali configurazioni si è proceduto al campionamento casuale di un punto, ottenendo infine il seguente workload sintetico di alto livello HL_c :

	timeStamp	elapsed	label	responseCode	responseMessage	threadName	dataType	success	failureMessage	bytes	sentBytes	grpThreads	allThreads	URL	Latency	IdleTime	Connect	
19	160819966098	111	HTTP Request small	200		OK Thread Group 1 1-5	bin	True		Nan	152918	133	5	9	http://192.168.56.101/jupiter_small.jpg	103	0	100
102	160819962139	337	HTTP Request small	200		OK Thread Group 2 2-11	bin	True		Nan	152918	133	11	39	http://192.168.56.101/jupiter_small.jpg	331	0	4
550	1608200084085	22	HTTP Request mid	200		OK Thread Group 1 1-22	bin	True		Nan	1031745	133	30	131	http://192.168.56.101/cassiopea_mid.jpg	3	0	1
912	1608200179079	34	HTTP Request big	200		OK Thread Group 1 1-17	bin	True		Nan	2962330	133	30	134	http://192.168.56.101/andromeda_big.jpg	1	0	1
1052	1608200216095	7	HTTP Request small	200		OK Thread Group 1 1-30	bin	True		Nan	152918	133	30	101	http://192.168.56.101/jupiter_small.jpg	3	0	1

Figura 3.6: Workload sintetico HL_c

Workload Characterization LL

Un procedimento analogo è stato applicato anche per la raccolta ed analisi dei parametri di LL, i quali sono relativi al sistema server. Per collezionare i dati sul server è stata utilizzata l'utility *vmstat*, la quale permette di registrare il funzionamento della macchina. Con un tempo di campionamento pari ad 1 secondo ed un tempo totale pari a 300 secondi, si è quindi registrato il comportamento del server Apache in contemporanea alla somministrazione di richieste lato client con Jmeter.

Si riporta un estratto del workload reale di basso livello:

r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st	
0	0	0	31884	117052	30156	631016	0	0	0	4	150	471	6	0	94	0	0
1	1	0	31884	117052	30168	631020	0	0	0	96	152	386	5	1	94	0	0
2	0	0	31884	117052	30168	631020	0	0	0	4	132	372	7	1	92	0	0
3	1	0	31884	117052	30168	631020	0	0	0	4	140	364	5	2	93	0	0
4	1	0	31884	117052	30168	631020	0	0	0	8	137	359	5	0	95	0	0
...
295	0	0	31884	106896	30624	631472	0	0	20	120	637	486	6	8	86	0	0
296	0	0	31884	106896	30624	631480	0	0	20	4	483	463	9	9	81	0	0
297	0	0	31884	106896	30624	631480	0	0	20	4	196	382	5	3	91	1	0
298	1	0	31884	106896	30624	631480	0	0	0	0	602	356	4	7	89	0	0
299	0	0	31884	106896	30624	631480	0	0	20	4	128	361	4	2	94	0	0

Figura 3.7: Workload LL

In prima battuta, sono state eliminate le feature costanti b , $swpd$, si , so , st , in quanto non contribuiscono alla varianza del dataset.

Dalla PCA condotta, si è scelto di mantenere 6 componenti principali, spiegando il 96.4% della varianza totale.

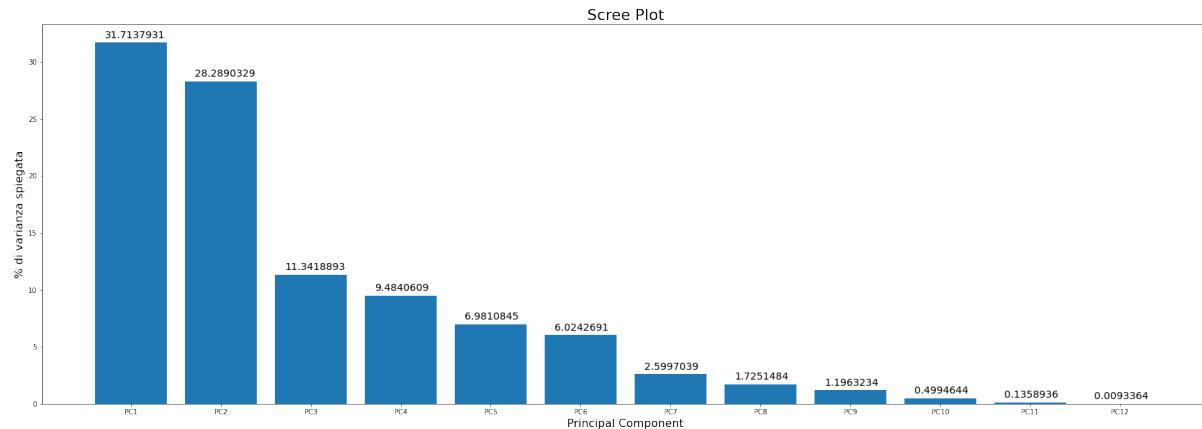


Figura 3.8: Workload LL

Dalla fase di Clustering ne risulta invece come 9 cluster sia il miglior compromesso tra devianza persa e riduzione della dimensione del dataset.

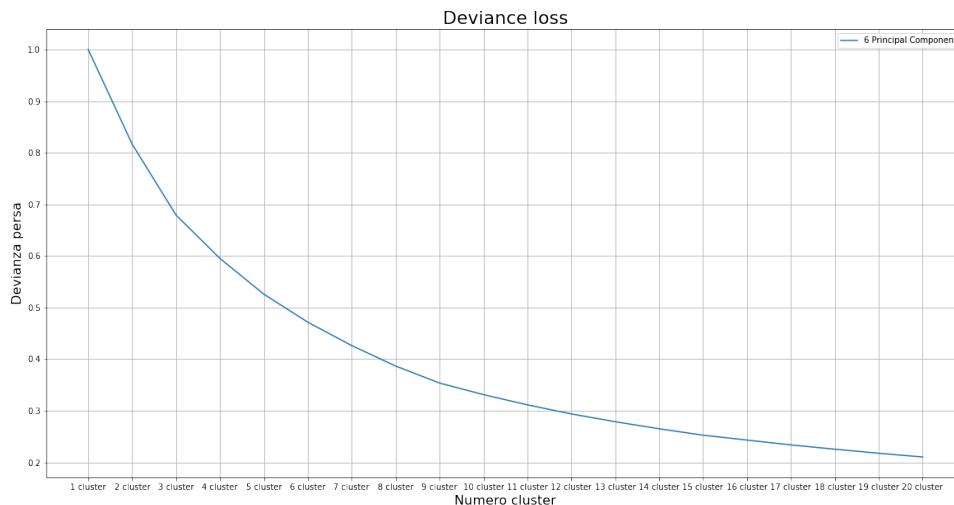


Figura 3.9: Clustering - Devianza persa

La percentuale di devianza mantenuta al termine delle analisi effettuate è pari al 65%. La Workload Characterization del workload di LL è stata ultimata con il campionamento casuale di un punto per ogni cluster.

3.3.2 Fase 2: Generazione di LL'_c a partire da HL_c

Il medesimo procedimento è stato ripetuto ancora una volta per estrarre un nuovo workload di basso livello LL' , stavolta ottenuto registrando l'attività del sistema server con la somministrazione di un nuovo test condotto sulla base dei risultati ottenuti durante la caratterizzazione del workload di alto livello HL . Le combinazioni utilizzate sono di fatto *small-1* (x2), *mid-1*, *big-1*, *small-2*.

Tale workload è stato caratterizzato effettuando le stesse scelte precedenti in merito a numero di componenti principali (6) e numero di cluster (9), permettendo la generazione del workload sintetico LL'_c .

3.3.3 Fase 3: Validazione di LL'_c

Per ultimare l'analisi, è necessario andare a confrontare il workload di basso livello LL_c , ottenuto a partire dal workload reale di alto livello, con il workload di basso livello LL'_c , ottenuto a partire dal workload sintetico di alto livello. Se tali workload non risultano essere differenti statisticamente, possiamo concludere che il workload sintetico di alto livello HL_c è statisticamente rappresentativo del workload reale di alto livello HL .

Innanzitutto, si è verificato se i dati seguissero una distribuzione normale, in modo da poter poi eseguire un test parametrico per la validazione di LL'_c . Si è eseguito in prima battuta un test visivo riportato di seguito, basato sui grafici Quantile-Quantile (**QQ Plot**), in modo da confrontare per ognuna delle colonne se essa seguisse o meno una distribuzione normale. Tale procedimento è stato effettuato sia per LL_c che per LL'_c , e non sembra evidenziare per tutte le colonne l'ipotesi avanzata.

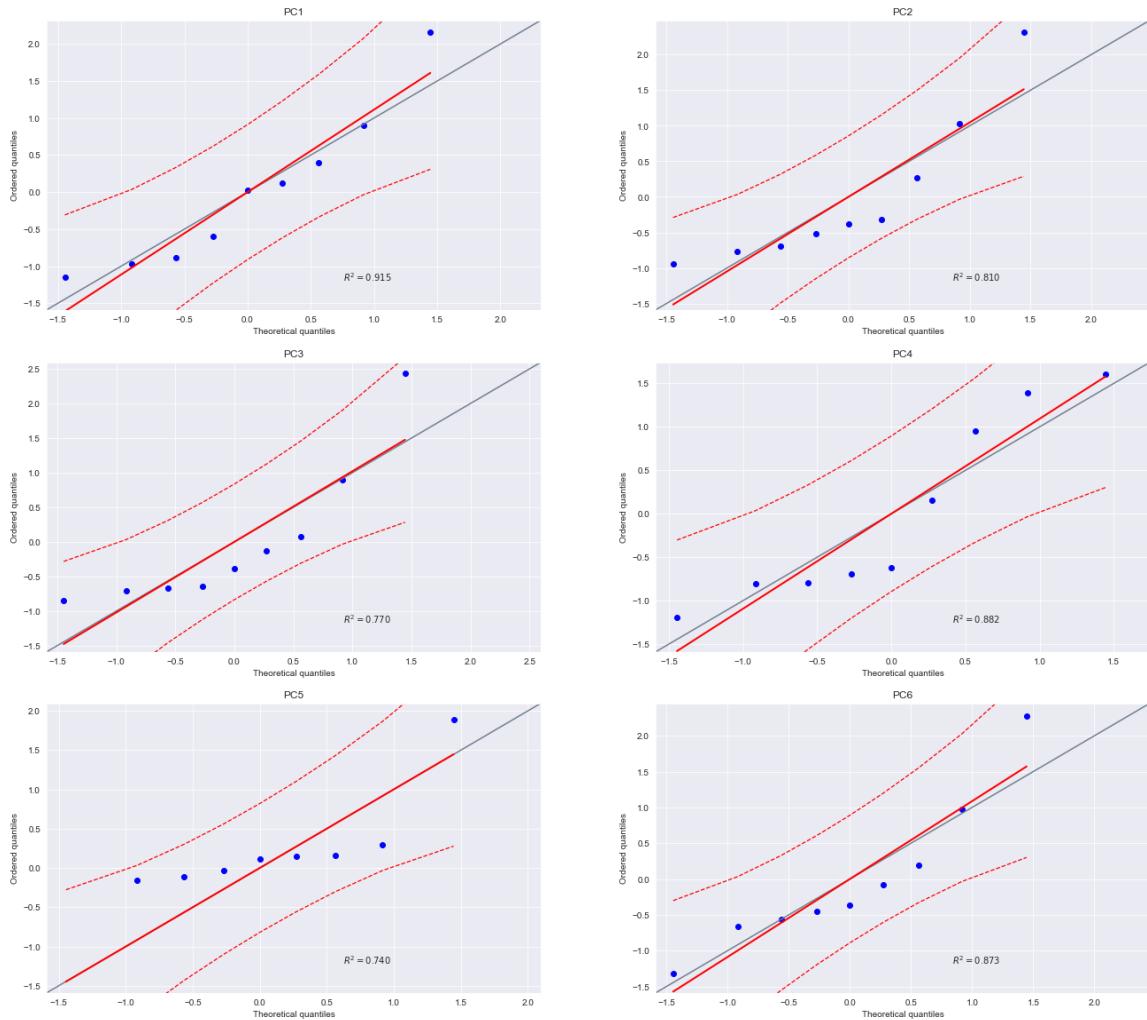


Figura 3.10: QQ Plot LL_c

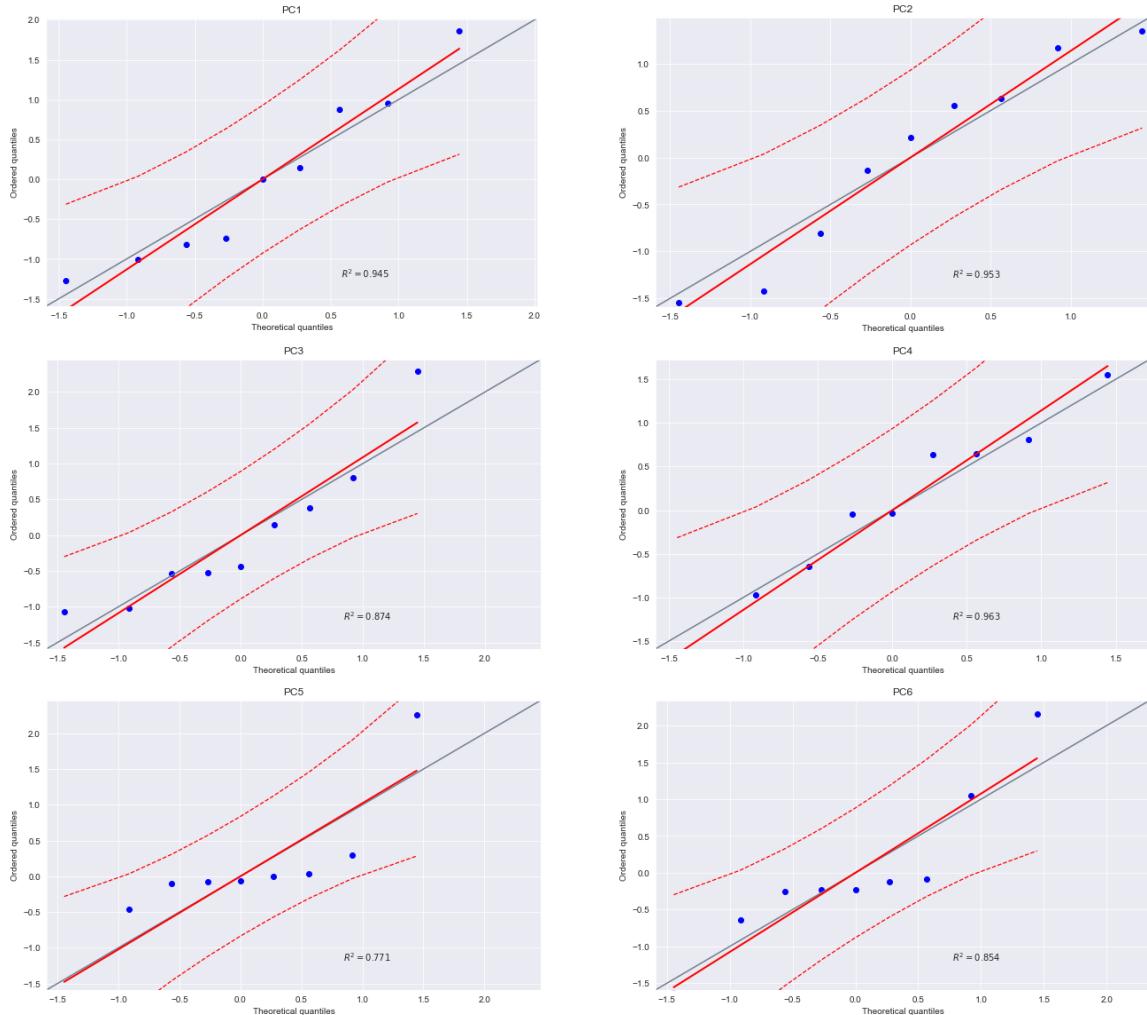


Figura 3.11: QQ Plot LL'_c

L'intuizione visiva è confermata dal test di Shapiro-Wilk, che non fornisce risultato affermativo per tutte le colonne in gioco; l'ipotesi di normalità, quindi, non è rispettata da nessuno dei due workload in esame, con un livello di confidenza del 95%.

LL'_c	PC	R^2	p-value	Normalità
	PC1	0.914853	0.351333	True
	PC2	0.817698	0.032454	False
	PC3	0.78148	0.012526	False
	PC4	0.860192	0.096347	True
	PC5	0.795248	0.01802	False
	PC6	0.891121	0.204885	True

PC	R^2	p-value	Normalità
PC1	0.93373	0.517648	True
PC2	0.928592	0.46806	True
LL'_c	0.880354	0.158372	True
PC4	0.968239	0.879261	True
PC5	0.823752	0.037986	False
PC6	0.88347	0.170739	True

Si è proceduto quindi ad effettuare il test non parametrico noto come *Somma dei ranghi*, o *test di Wilcoxon*. Tale test è stato eseguito per ogni coppia di colonne dei due workload, con l'obiettivo di verificare se il *p-value* risultante dalla somma dei ranghi sia maggiore del *p-value* di riferimento fissato, $p-value_{rif} = 0.05$, in modo da poter affermare che non ci siano differenze statistiche tra i dataset ed accettare di conseguenza l'ipotesi nulla H_0 secondo cui i dati provengano dalla stessa distribuzione. L'ipotesi nulla è stata accettata per ogni coppia di colonne, da cui si può concludere come il workload LL'_c non presenti differenze statistiche significative con il workload LL_c . Ne consegue che il workload sintetico di alto livello HL_c sia rappresentativo di quello reale HL .

PC-PC	W-val	tail	p-value	RBC	CLES
PC1	11.0	two-sided	0.203125	0.511111	0.518519
PC2	13.0	two-sided	0.300781	0.422222	0.54321
PC3	18.0	two-sided	0.652344	-0.2	0.333333
PC4	18.0	two-sided	0.652344	0.2	0.555556
PC5	20.0	two-sided	0.820312	0.111111	0.45679
PC6	17.0	two-sided	0.570312	0.244444	0.716049

3.4 Capacity Test

Il Capacity Test rientra nell'ambito delle Performance Analysis in quanto mira a caratterizzare le performance del sistema sotto differenti condizioni di lavoro. L'obiettivo di tale studio consiste nell'identificare i limiti di funzionamento del sistema.

Nel caso del Web Server, le metriche tipicamente utilizzate per caratterizzare le performance sono:

- **Response time:** intervallo di tempo che intercorre tra la richiesta di una risorsa e il completamento della risposta del sistema.
- **Throughput:** il numero di richieste per unità di tempo che il server è in grado di soddisfare.

Nel contesto di questo esercizio, il capacity test è stato eseguito con diverse configurazioni di richieste lato client:

- Richieste **Small:** il client effettua solo richieste di tipo Small
- Richieste **Big:** il client effettua solo richieste di tipo Big
- Richieste **Random:** il client effettua tutte le richieste (Small, Medium, Big) in maniera randomica.

Per ogni livello di carico considerato sono state eseguite 5 ripetizioni della durata di 1 minuto ciascuna. Per ogni ripetizione, sono stati calcolati il throughput e il response time medio: il throughput è pari al numero di richieste servite correttamente dal server (non andate in errore) al secondo, mentre il response time è stato calcolato come il valore medio della colonna *elapsed*, e non *latency*, in quanto si preferisce misurare l'intervallo tra la fine della richiesta e la fine della risposta, in modo da poter capire come varia il comportamento del sistema in relazione alla grandezza della risorsa richiesta.

Per ogni valore di carico, si è poi calcolato il valore medio tra le 5 ripetizioni effettuate, in modo da avere un unico indicatore di throughput e response time per ogni livello di carico considerato.

A questo punto, sono stati realizzati i grafici di Throughput, Response Time e Potenza in relazione al carico, in modo da poter individuare i punti di lavoro del sistema:

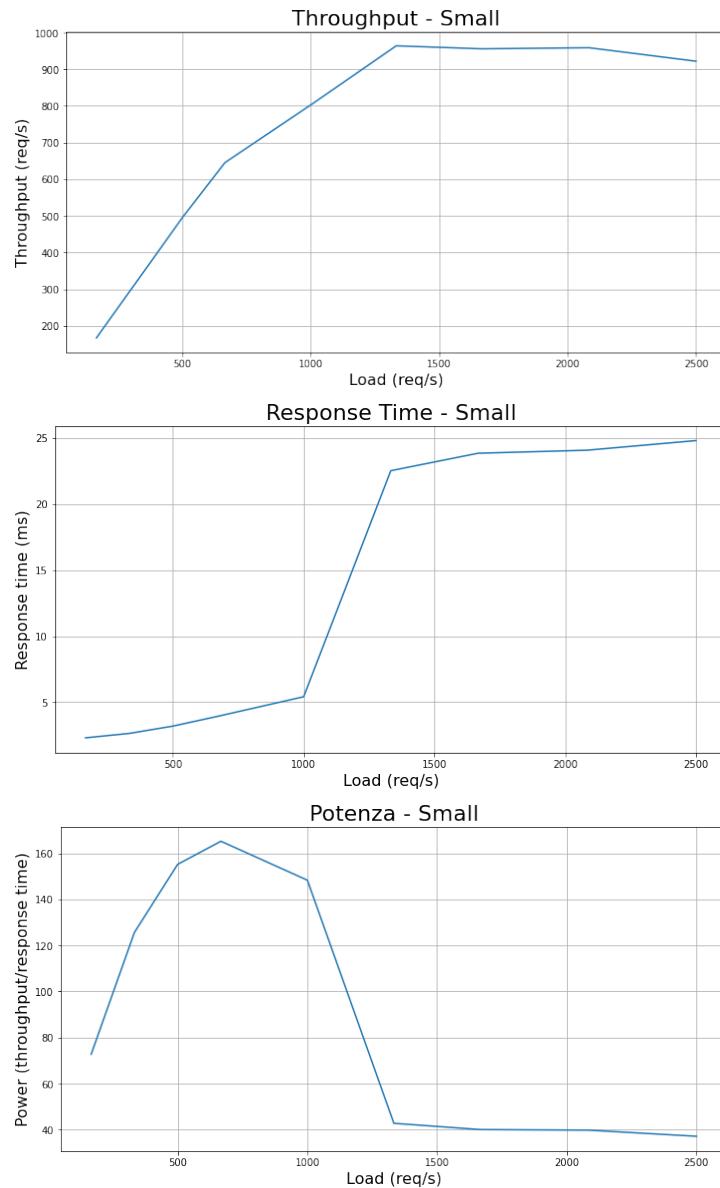
- **Knee Capacity:** punto di lavoro in cui il sistema ha un buon compromesso tra throughput (alto) e tempo di risposta (basso). Ha la peculiarità di essere il punto in cui la potenza assume il valore massimo. Esso rappresenta un buon livello di carico su cui far lavorare il sistema.
- **Usable Capacity:** punto di lavoro in cui si registra il throughput massimo raggiungibile dal sistema, senza superare uno specifico limite di tempo di risposta. Rappresenta il punto limite di funzionamento del sistema, oltre il quale il throughput in genere si riduce drasticamente.

È bene osservare che, nei vari capacity test condotti nel contesto di questo esercizio, il massimo carico imposto al sistema è stato individuato nel valore di carico in cui il sistema server presentasse una percentuale di errore del 10-15%.

3.5 Risorsa Small

Per il capacity test con richiesta di risorse di tipo Small, si è utilizzato un unico Thread Group con 30 threads attivi. Il carico imposto al sistema varia tra 10000 richieste al minuto (167 req/s) e 150000 richieste al minuto (2500 req/s).

Di seguito si riportano i grafici di Throughput, Response Time e Potenza:



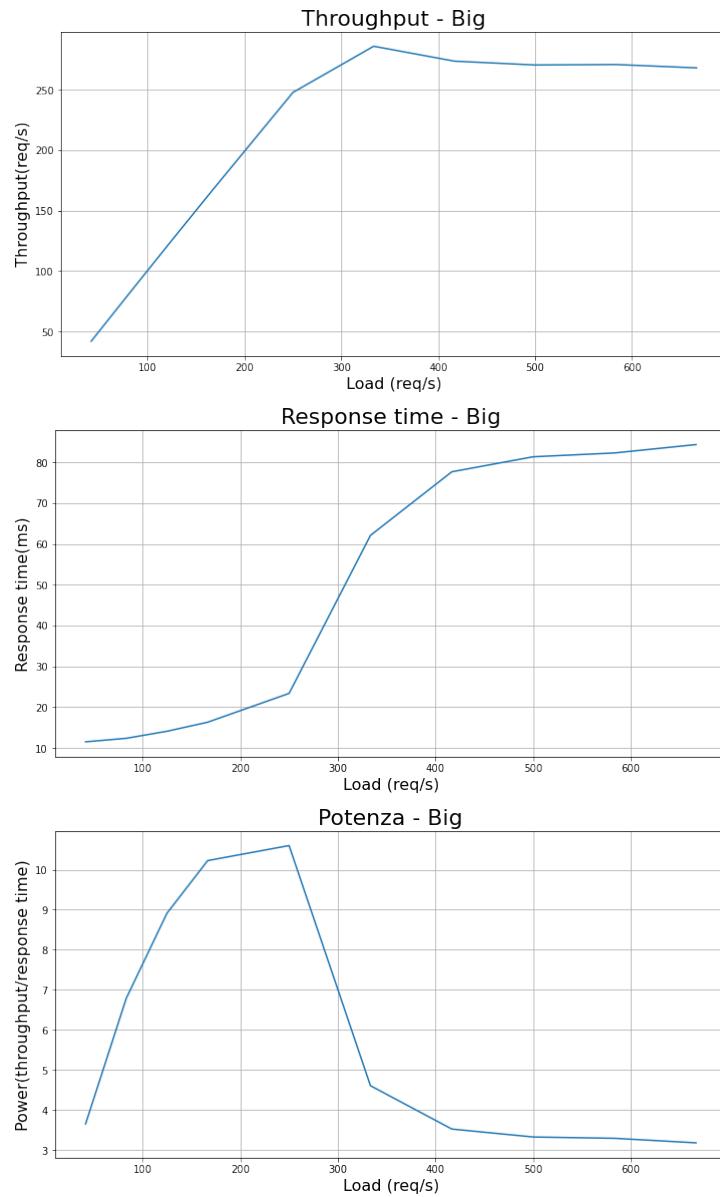
Per tale configurazione i punti di lavoro scelti sono i seguenti:

- **Knee Capacity:** 666 req/s
- **Usable Capacity:** 2000 req/s

3.6 Risorsa Big

Per il capacity test con richiesta di risorse di tipo Big, si è utilizzato un unico Thread Group con 30 threads attivi. Il carico imposto al sistema varia tra 2500 richieste al minuto (42 req/s) e 40000 richieste al minuto (666 req/s).

Di seguito si riportano i grafici di Throughput, Response Time e Potenza:



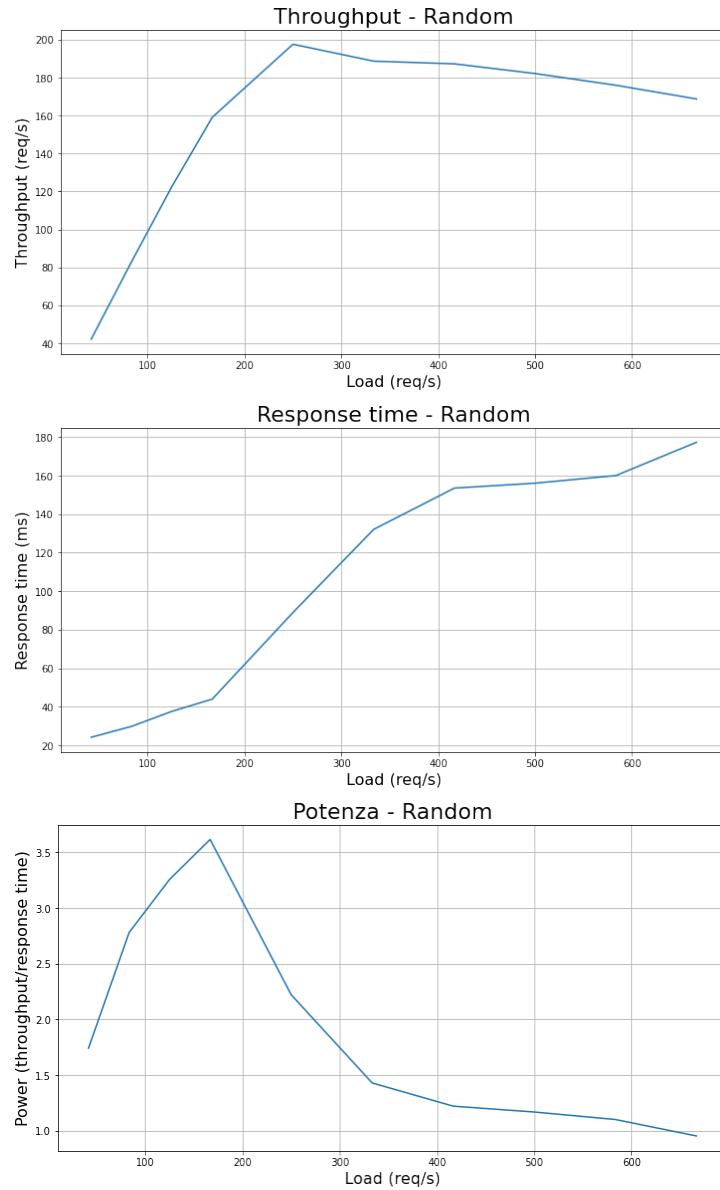
Per tale configurazione i punti di lavoro scelti sono i seguenti:

- **Knee Capacity:** 250 req/s
- **Usable Capacity:** 600 req/s

3.7 Risorse Random

Per il capacity test con richiesta di risorse di tutte le tipologie (small, medium e big), si è utilizzato un unico Thread Group con 30 threads attivi. Il carico imposto al sistema varia tra 2500 richieste al minuto (42 req/s) e 40000 richieste al minuto (666 req/s).

Di seguito si riportano i grafici di Throughput, Response Time e Potenza:



Per tale configurazione i punti di lavoro scelti sono i seguenti:

- **Knee Capacity:** 166 req/s
- **Usable Capacity:** 500 req/s

3.8 Design of Experiment (DoE)

L'ultimo studio proposto è quello del *Design of Experiment*, il quale si propone di verificare l'infuenza dei fattori sul response time del sistema. Nel contesto di tale esercizio, sono stati scelti 2 fattori differenti, ognuno dei quali può assumere 3 valori (livelli) differenti:

- **Load**(carico) imposto al sistema server da parte del client, espresso in termini di richieste al secondo. I 3 livelli sono stati ottenuti prendendo come riferimento la media dei valori di Usable Capacity ottenuti nei 3 capacity test realizzati nel precedente esercizio, pari a 1033 req/s . Sulla base di essi, sono stati individuati 3 differenti livelli di carico nel seguente modo:
 - Light: 25% della Usable Capacity (250 req/s)
 - Medium: 50% della Usable Capacity (500 req/s)
 - Heavy: 75% della Usable Capacity (750 req/s)
- **Tipologia di risorsa** richiesta (Small, Medium, Big)

Sulla base di tali fattori e livelli, si è scelto di adottare un **full-factorial design** con 5 ripetizioni per ogni combinazione, che ha portato alla conduzione di $3^2 \cdot 5 = 45$ esperimenti.

Si è deciso effettuare anche delle repliche per scindere il contributo dell'errore dal contributo dell'interazione dei fattori.

Si riportano i risultati degli esperimenti condotti:

Carico	Risorsa	Tempi risposta
250 req/s	small	2.60
250 req/s	small	2.27
250 req/s	small	2.11
250 req/s	small	1.99
250 req/s	small	3.07
500 req/s	small	11.27
500 req/s	small	3.18
500 req/s	small	2.83
500 req/s	small	2.90
500 req/s	small	12.15
750 req/s	small	10.88
750 req/s	small	12.55
750 req/s	small	10.64
750 req/s	small	11.88
750 req/s	small	13.01
250 req/s	medium	8.91
250 req/s	medium	9.28
250 req/s	medium	7.14
250 req/s	medium	11.14
250 req/s	medium	8.03
500 req/s	medium	64.30
500 req/s	medium	68.34
500 req/s	medium	62.10
500 req/s	medium	78.13
500 req/s	medium	80.09
750 req/s	medium	87.22
750 req/s	medium	84.46
750 req/s	medium	80.11
750 req/s	medium	74.24
750 req/s	medium	72.09
250 req/s	big	14.76
250 req/s	big	19.66
250 req/s	big	13.82
250 req/s	big	18.08
250 req/s	big	15.11
500 req/s	big	82.27
500 req/s	big	78.49
500 req/s	big	82.87
500 req/s	big	79.80
500 req/s	big	83.35
750 req/s	big	86.41
750 req/s	big	98.11
750 req/s	big	84.30
750 req/s	big	81.33
750 req/s	big	86.22

Effetti dei livelli dei fattori

Il modello della risposta del sistema è il seguente:

$$y_{ijk} = \mu + \alpha_j + \beta_i + \gamma_{ij} + e_{ijk}$$

- y_{ijk} è l'osservazione con il fattore carico a livello j e il fattore risorsa al livello i durante la k-esima ripetizione.
- μ è la risposta media.
- α_j è l'effetto del fattore carico a livello j.
- β_i è l'effetto del fattore risorsa a livello i.
- γ_{ij} è l'effetto dell'interazione tra il fattore carico a livello i e il fattore risorsa a livello j.
- e_{ijk} è l'errore sperimentale.

Sfruttando le proprietà di somma degli effetti, delle interazioni e dell'errore nulli, sono stati determinati gli effetti dei livelli di ciascun fattore:

- $\alpha_j = \bar{y}_{.j.} - \bar{y}_{...}$
- $\beta_i = \bar{y}_{i..} - \bar{y}_{...}$
- $\gamma_{ij} = \bar{y}_{ij.} - \bar{y}_{i..} - \bar{y}_{.j.} + \bar{y}_{...}$

Calcolo degli errori

La risposta stimata dal modello è:

$$\hat{y}_{ij} = \mu + \alpha_j + \beta_i + \gamma_{ij}$$

Utilizzando questa risposta stimata \hat{y}_{ij} e la risposta osservata y_{ijk} è possibile definire l'errore come:

$$e_{ij} = y_{ijk} - \hat{y}_{ij}$$

Allocazione della varianza: Importanza

A questo punto è necessario calcolare l'importanza dei due fattori carico e risorsa, della loro interazione e dell'errore. Essa può essere calcolata come il rapporto tra la devianza del fattore e la devianza totale delle risposte osservate:

$$SST = \sum_{i=1}^3 \sum_{j=1}^3 \sum_{k=1}^5 (y_{ijk} - \bar{y}_{...})^2$$

$$SSA = 3 \cdot 5 \cdot \sum_{j=1}^3 \alpha_j^2$$

$$SSB = 3 \cdot 5 \cdot \sum_{i=1}^3 \beta_i^2$$

$$SSAB = 5 \cdot \sum_{i=1}^3 \sum_{j=1}^3 \gamma_{ij}^2$$

$$SSE = \sum_{i=1}^3 \sum_{j=1}^3 \sum_{k=1}^5 e_{ijk}^2$$

Analisi della varianza					
Origine	DF	Somma dei quadrati	Media quadratica	Rapporto F	
Modello	8	56557,860	7069,73	342,7357	
Errore	36	742,585	20,63	Prob > F	
C. totale	44	57300,445		<,0001*	

Stime dei parametri					
Test degli effetti					
Origine	Nparm	DF	Somma dei quadrati	Rapporto F	Prob > F
Carico	2	2	22419,439	543,4393	<,0001*
Risorsa	2	2	26006,725	630,3939	<,0001*
Carico*Risorsa	4	4	8131,696	98,5547	<,0001*

Figura 3.15: Analisi della varianza

Possiamo quindi calcolare l'importanza dei fattori (A=carico, B=risorsa), dell'interazione (SSAB) e dell'errore (SSE):

$$\frac{SSA}{SST} = \frac{22419.439}{57300.445} = 0.391$$

$$\frac{SSB}{SST} = \frac{26006.725}{57300.445} = 0.454$$

$$\frac{SSAB}{SST} = \frac{8131.696}{57300.445} = 0.142$$

$$\frac{SSE}{SST} = \frac{742.585}{57300.445} = 0.013$$

Il fattore B, cioè il fattore risorsa, risulta essere il fattore più importante. Inoltre si può osservare come entrambi i fattori siano nettamente più importanti della loro interazione, e l'errore è il meno importante.

Allocazione della varianza: Significatività

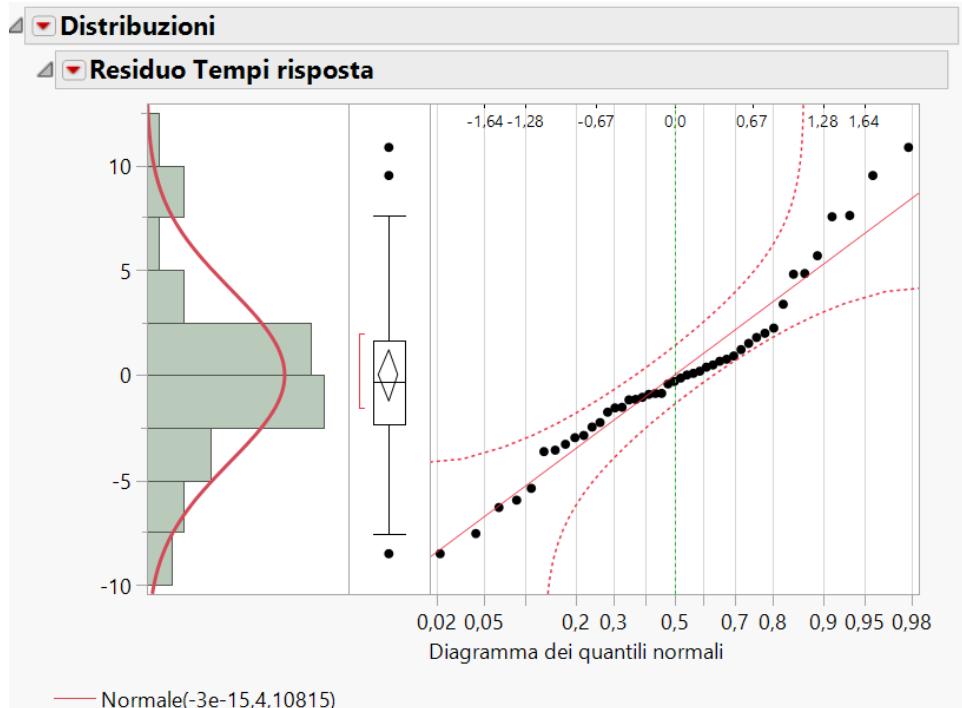
Prima di procedere con lo studio della significatività, è necessario analizzare due assunzioni fondamentali, ossia la normalità e l'omoschedasticità dei residui dei tempi di risposta. Tali ipotesi vanno verificate necessariamente per definire la tipologia di test da poter effettuare, come risulta dalla seguente tabella:

Normalità	Omoschedasticità	Test ANOVA
✓	✓	<i>F-Test</i>
✗	✓	<i>Kruskal-Wallis test</i>
✓	✗	<i>Welch test</i>
✗	✗	<i>Kruskal-Wallis / Friedman test</i>

In prima battuta sono stati calcolati proprio i residui per i vari esperimenti a partire dai tempi di risposta.

Carico	Risorsa	Tempi risposta	Residuo
250 req/s	small	2,6	0,192
250 req/s	small	2,27	-0,138
250 req/s	small	2,11	-0,298
250 req/s	small	1,99	-0,418
250 req/s	small	3,07	0,662
500 req/s	small	11,27	4,804
500 req/s	small	3,18	-3,286
500 req/s	small	2,83	-3,636
500 req/s	small	2,9	-3,566
500 req/s	small	12,15	5,684
750 req/s	small	10,88	-0,912
750 req/s	small	12,55	0,758
750 req/s	small	10,64	-1,152
750 req/s	small	11,88	0,088
750 req/s	small	13,01	1,218
250 req/s	medium	8,91	0,01
250 req/s	medium	9,28	0,38
250 req/s	medium	7,14	-1,76
250 req/s	medium	11,14	2,24
250 req/s	medium	8,03	-0,87
500 req/s	medium	64,3	-6,292
500 req/s	medium	68,34	-2,252
500 req/s	medium	62,1	-8,492
500 req/s	medium	78,13	7,538
500 req/s	medium	80,09	9,498
750 req/s	medium	87,22	7,596
750 req/s	medium	84,46	4,836
750 req/s	medium	80,11	0,486
750 req/s	medium	74,24	-5,384
750 req/s	medium	72,09	-7,534
250 req/s	big	14,76	-1,526
250 req/s	big	19,66	3,374
250 req/s	big	13,82	-2,466
250 req/s	big	18,08	1,794
250 req/s	big	15,11	-1,176
500 req/s	big	82,27	0,914
500 req/s	big	78,49	-2,866
500 req/s	big	82,87	1,514
500 req/s	big	79,8	-1,556
500 req/s	big	83,35	1,994
750 req/s	big	86,41	-0,864
750 req/s	big	98,11	10,836
750 req/s	big	84,3	-2,974
750 req/s	big	81,33	-5,944
750 req/s	big	86,22	-1,054

Si è proceduto alla verifica della normalità dei residui, effettuando in primo luogo il seguente test visivo tramite il diagramma dei quantili normali:



Quantili		
100.0%	massimo	10,836
99.5%		10,836
97.5%		10,6353
90.0%		6,4256
75.0%	quartile	1,654
50.0%	mediana	-0,298
25.0%	quartile	-2,359
10.0%		-5,608
2.5%		-8,3483
0.5%		-8,492
0.0%	minimo	-8,492

Statistiche di riepilogo		
Media		-2,57e-15
Dev std		4,1081544
Errore std della media		0,6124075
Media superiore al 95%		1,2342262
Media inferiore al 95%		-1,234226
N		45

Il test visivo permette di non rigettare l'ipotesi di normalità, in quanto tutti i residui sono contenuti nelle fasce individuate. Si è proceduto a verificare tale test tramite il test di Shapiro-Wilk, o test della bontà di adattamento. Tale test restituisce risultato affermativo, come si può notare dalla seguente immagine, permettendoci di fatto di confermare la normalità dei residui.

Stima normale

Stime dei parametri		Stima	Inferiore	Superiore
Tipo	Parametro		al 95%	al 95%
Percorso	μ	-2,57e-15	-1,234226	1,2342262
Dispersione	σ	4,1081544	3,4009531	5,1894192
Misura				
-2*Log verosimiglianza		253,87212		
AICc		258,15783		
BIC		261,48544		

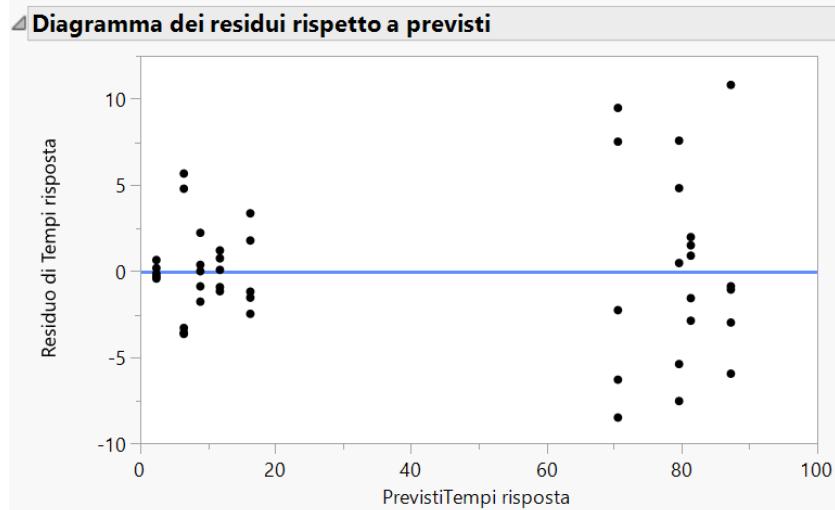
Test della bontà di adattamento

Test W di Shapiro-Wilk

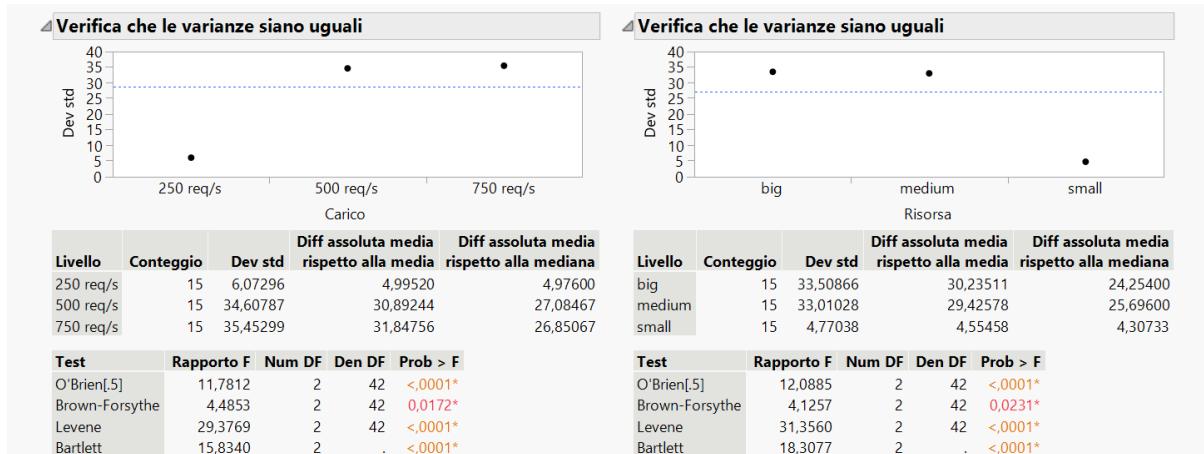
W	Prob<W
0,959019	0,1121

Nota: Ho = i dati provengono dalla distribuzione Normale. I p-value bassi rifiutano Ho.

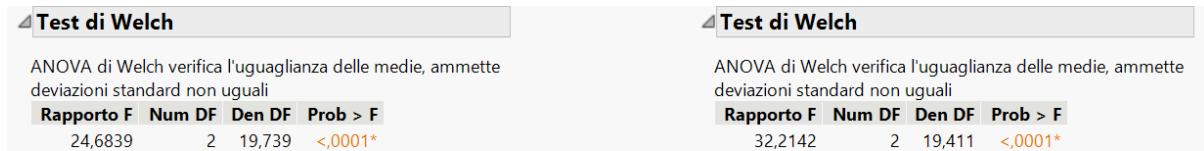
In secondo luogo è stata verificata l'omoschedasticità per ognuno dei fattori in gioco. Si riporta in primo luogo un test visivo, valutato graficando i residui dei tempi di risposta rispetto ai tempi di risposta stimati. Essendo la dispersione di tali punti non costante, è possibile intuire come in realtà i residui non siano omoschedastici.



Si è proceduto col confermare l'intuizione tramite un test analitico. JMP effettua in automatico la stima dell'omogeneità delle varianze tramite 5 test differenti, tra i quali si è scelto il test di Levene. Dai risultati del test (riportati di seguito) si evince come l'ipotesi nulla viene rigettata, dunque le varianze dei residui non sono omogenee.



Essendo verificata la normalità ma non l'omoschedasticità dei residui, si procede con un test parametrico ed eteroschedastico quale il Welch test, il quale restituisce esito positivo per entrambi i fattori e permette di concludere come entrambi i fattori siano statisticamente significativi con un livello di confidenza del 95%.



Es. 4 - Reliability

4.1 Esercizio 1

4.1.1 Traccia

Calcolare la Reliability $R(t)$ e il Mean Time To Fail (MTTF) per il sistema, il cui reliability diagram è fornito di seguito. Nel calcolo dell'MTTF, si presume che tutti i componenti siano identici e si guastino casualmente con tasso di errore λ .

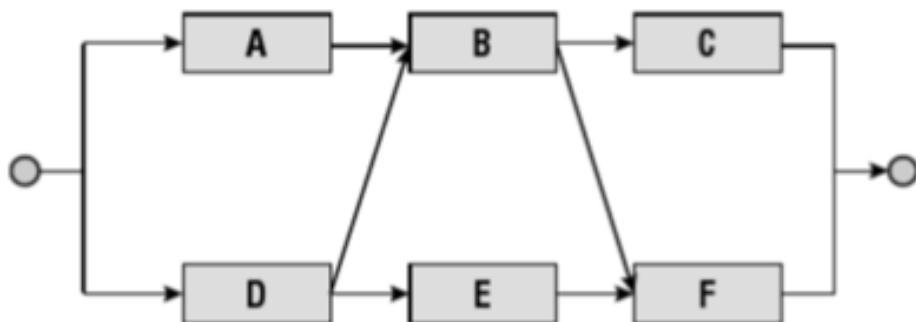


Figura 4.1: Reliability diagram - esercizio 1

4.1.2 Risoluzione

In questo esercizio è richiesto di studiare un RBD (reliability block diagram) di un sistema non scomponibile in serie e parallelo.

Una prima tecnica che possiamo applicare per la risoluzione è basata sul **Teorema dell'Upper Bound**, che ci permette di trasformare un reliability block diagram come un parallelo di serie dove ogni serie è un **success path**, ossia un path che va dalla sorgente alla destinazione passando per componenti diversi. Tramite esso si può dunque pervenire all'Upper Bound della Reliability $R(t)$ del sistema:

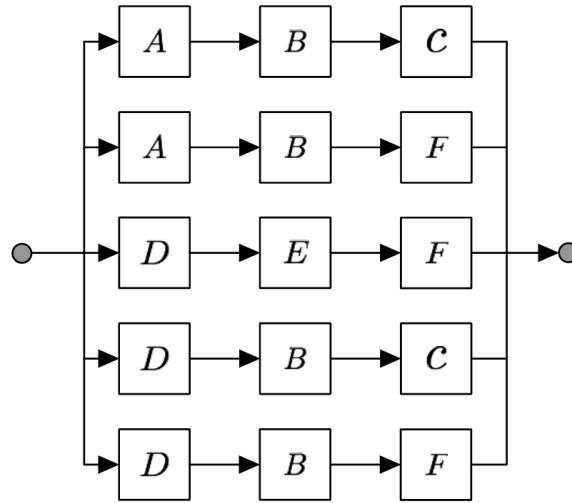


Figura 4.2: Upper Bound del problema

$$\begin{aligned}
 R_{sys} &\leq 1 - (1 - R_A R_B R_c) (1 - R_A R_B R_F) (1 - R_D R_E R_F) (1 - R_D R_B R_C) (1 - R_D R_B R_F) \\
 &= 1 - (1 - e^{-3\lambda t})^5
 \end{aligned}$$

Si noti che l'uguaglianza è valida solo se i success path sono indipendenti, cioè se non ci sono componenti in comune tra i vari path. Nel nostro caso, invece, dato che tale condizione non è verificata, tale valore è da considerarsi esclusivamente come un limite superiore. Quindi risulta che l'Upper Bound per la Reliability del sistema è:

$$R_{sys} < 1 - (1 - e^{-3\lambda t})^5$$

Per calcolare analiticamente la Reliability del sistema, è necessario ricorrere alla regola di Bayes, qui riportata:

$$\sum_{i=1}^n P(A | B_i) \cdot P(B_i) \quad \bigcup_{i=1}^n B_i = A \quad \bigcap_{i=1}^n B_i = \emptyset$$

A questo punto, utilizzando tale formula si può procedere con l'applicazione del **Conditioning**, che ci permette di scrivere la reliability totale del sistema, tenendo conto del condizionamento rispetto a un generico nodo M , nel seguente modo:

$$R_{sys} = R_M \cdot P(sys | M \text{ funziona}) + (1 - R_M) \cdot P(sys | M \text{ fallisce})$$

Si è scelto allora di applicare il conditioning rispetto al nodo E:

Caso 1 - E fallisce:

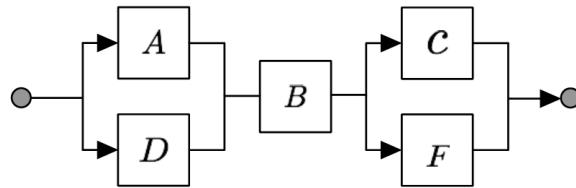


Figura 4.3: Conditioning rispetto a E che fallisce

$$\begin{aligned} P(sys \mid E \text{ fallisce}) &= (1 - (1 - R_A)(1 - R_D)) R_B (1 - (1 - R_C)(1 - R_F)) \\ &= e^{-\lambda t} \left(1 - (1 - e^{-\lambda t})^2\right)^2 \end{aligned}$$

Caso 2 - E funziona:

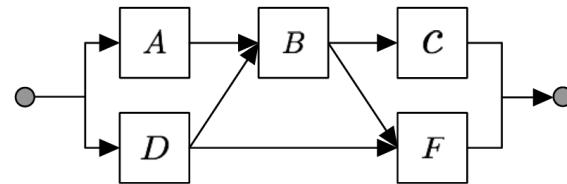


Figura 4.4: Upper Bound del problema

In questo caso è necessario riapplicare il conditioning rispetto al nodo B:

Caso 2.1 - E funziona, B fallisce:

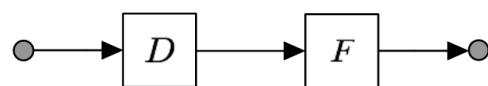


Figura 4.5: Upper Bound del problema

$$P(sys \mid E \text{ funziona}, B \text{ fallisce}) = R_D R_E = e^{-2\lambda t}$$

Caso 2.2 - E funziona, B funziona:

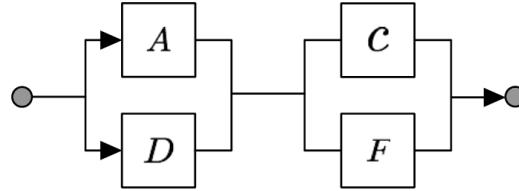


Figura 4.6: Upper Bound del problema

$$\begin{aligned} P(\text{sys} \mid E \text{ funziona}, B \text{ funziona}) &= (1 - (1 - R_A)(1 - R_D))(1 - (1 - R_C)(1 - R_F)) \\ &= \left(1 - (1 - e^{-\lambda t})^2\right)^2 \end{aligned}$$

Con le probabilità condizionate appena calcolate, si può procedere al calcolo della Reliability del sistema:

$$\begin{aligned} R_{sys} &= P(\text{sys} \mid E \text{ fallisce})(1 - R_E) + P(\text{sys} \mid E \text{ funziona}) \cdot R_E \\ &= P(\text{sys} \mid E \text{ fallisce})(1 - R_E) + P(\text{sys} \mid E \text{ funziona}, B \text{ fallisce}) \cdot R_E (1 - R_B) + \\ &\quad + P(\text{sys} \mid E \text{ funziona}, B \text{ funziona}) \cdot R_E R_B \\ &= e^{-\lambda t} \left(1 - (1 - e^{-\lambda t})^2\right)^2 \cdot (1 - e^{-\lambda t}) + e^{-3\lambda t} (1 - e^{-\lambda t}) + \left(1 - (1 - e^{-\lambda t})^2\right)^2 \cdot e^{-2\lambda t} \\ &= e^{-\lambda t} (e^{-2\lambda t} - 2e^{-\lambda t})^2 \cdot (1 - e^{-\lambda t}) + e^{-3\lambda t} (1 - e^{-\lambda t}) + (e^{-2\lambda t} - 2e^{-\lambda t})^2 \cdot e^{-2\lambda t} \\ &= e^{-\lambda t} (e^{-4\lambda t} - 4e^{-3\lambda t} + 4e^{-2\lambda t}) (1 - e^{-\lambda t}) + e^{-3\lambda t} - e^{-4\lambda t} + \\ &\quad + e^{-2\lambda t} (e^{-4\lambda t} - 4e^{-3\lambda t} + 4e^{-2\lambda t}) \\ &= (e^{-5\lambda t} - 4 \cdot e^{-4\lambda t} + 4 \cdot e^{-3\lambda t}) (1 - e^{-\lambda t}) + e^{-3\lambda t} - e^{-4\lambda t} + e^{-6\lambda t} - 4 \cdot e^{-5\lambda t} + 4 \cdot e^{-4\lambda t} \\ &= e^{-5\lambda t} - 5e^{-4\lambda t} + 5e^{-3\lambda t} \end{aligned}$$

In conclusione, la Reliability è pari a:

$$R_{sys}(t) = e^{-5\lambda t} - 5e^{-4\lambda t} + 5e^{-3\lambda t}$$

Integrando la Reliability, si può finalmente ottenere il **Mean Time To Fail** del sistema:

$$\begin{aligned} MTTF_{sys} &= \int_0^{+\infty} R_{sys}(t) dt = \int_0^{+\infty} e^{-5\lambda t} dt - 5 \int_0^{+\infty} e^{-4\lambda t} dt + 5 \int_0^{+\infty} e^{-3\lambda t} dt \\ &= -\frac{1}{5\lambda} [e^{-5t}]_0^{+\infty} + \frac{5}{4\lambda} [e^{-4t}]_0^{+\infty} - \frac{5}{3\lambda} [e^{-3t}]_0^{+\infty} = \frac{1}{5\lambda} - \frac{5}{4\lambda} + \frac{5}{3\lambda} = \frac{0.62}{\lambda} \end{aligned}$$

In conclusione, il **MTTF** del sistema è:

$$MTTF_{sys} = \frac{0.62}{\lambda}$$

4.2 Esercizio 2

4.2.1 Traccia

Vogliamo confrontare due diversi schemi di aumento della reliability di un sistema che utilizza la ridondanza. Supponiamo che il sistema necessiti di componenti identici in serie per il corretto funzionamento. Supponiamo inoltre che ci siano dati $m \times s$ componenti. Dato che la reliability di un singolo componente è r , derivare le espressioni per la reliability di due configurazioni. Per $m = 2$ e $s = 4$, confrontare le due espressioni. Dei due schemi mostrati nella figura sotto, quale fornisce una maggiore reliability? Modificare lo schema che ha minore affidabilità in modo da ottenere la stessa reliability dell'altro.

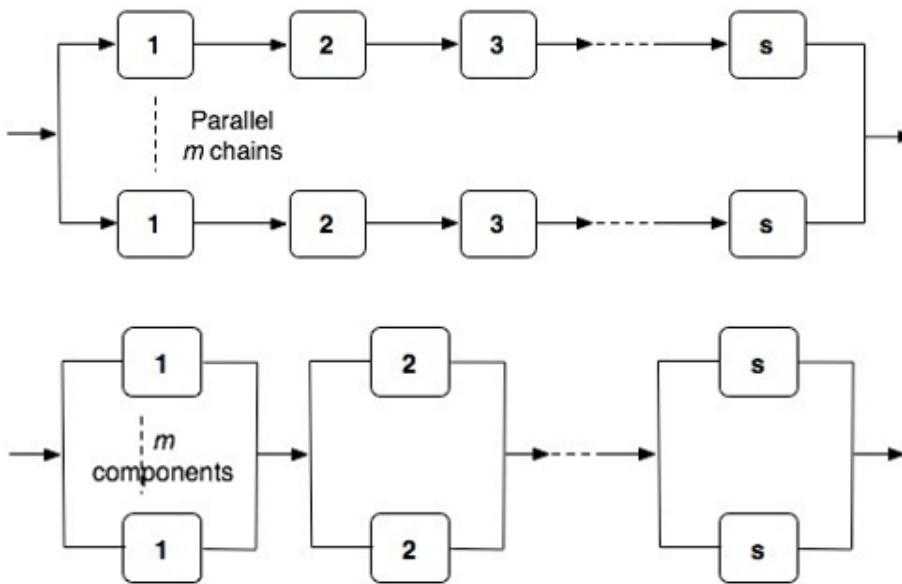


Figura 4.7: Traccia esercizio 2

4.2.2 Risoluzione

Essendo i due sistemi in esame costituiti da strutture note del tipo serie - parallelo, è possibile calcolare agevolmente le reliability di essi:

- Sistema S1: m paralleli di s serie, da cui risulta che la Reliability è pari a:

$$R_{S1} = 1 - \prod_1^m (1 - r^s) = 1 - (1 - r^s)^m$$

- Sistema S2: s serie di m paralleli, per cui la Reliability è:

$$R_{S2} = \left(1 - \prod_1^m (1 - r) \right)^s = (1 - (1 - r)^m)^s$$

Si può dedurre già dalla configurazione dei due schemi come in realtà il sistema S2 presenti una reliability superiore a quella del sistema S1, in quanto esso possiede un numero ben maggiore di success path dalla sorgente alla destinazione:

- il sistema S1 presenta semplicemente m success path differenti
- il sistema S2 presenta m^s differenti success path

Si può adesso procedere con il confronto grafico delle reliability dei due sistemi. Posti $m = 2$ e $s = 4$, posta $r = e^{-\lambda \cdot t}$ con $\lambda = 0.001$, per confrontare analiticamente le due espressioni si è tracciato un diagramma delle funzioni di reliability al variare del tempo (reliability dei singoli componenti):

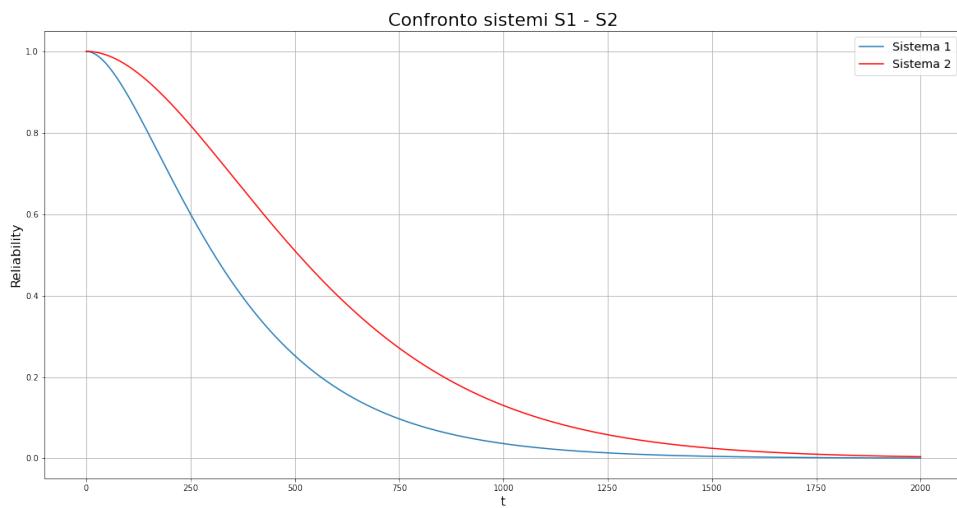


Figura 4.8: Confronto sistemi S1 e S2

La rappresentazione grafica conferma quanto ipotizzato precedentemente, poichè si nota come la curva di reliability del sistema S2 (in rosso) sia completamente al di sopra della curva di reliability del sistema S1 (in blu).

Per rendere il sistema S1 reliable quanto il sistema S2, è necessario modificarne la struttura in modo tale che i due sistemi abbiano lo stesso numero di success path, inizialmente rispettivamente m e m^s . Quindi è necessario modificare il parametro m per il sistema S1 in modo tale che esso sia pari a $m' = m^s$. La reliability di S1 sarà allora pari a:

$$R_{S1} = 1 - \prod_1^m (1 - r^s) = 1 - (1 - r^s)^{m^s}$$

Tale soluzione porta di fatto a una riconfigurazione del sistema S1 e comporta un aumento del numero di componenti utilizzati.

Si noti infine come tale ragionamento porta ad eguagliare gli upper bound dei due sistemi, ed in particolare l'upper bound per S1 sarà proprio pari alla reliability del sistema stesso essendo costituito da success path indipendenti (cosa che per S2 non avviene).

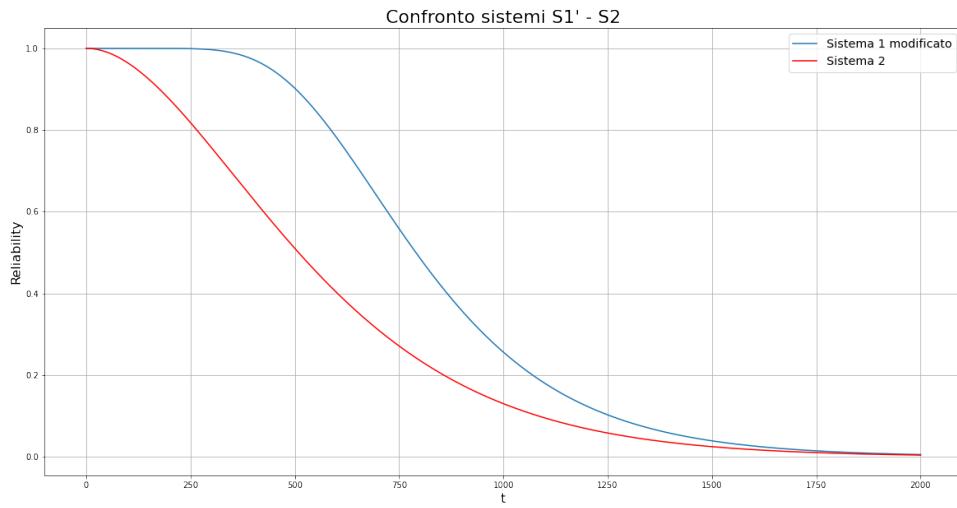


Figura 4.9: Confronto sistemi S1' e S2

4.3 Esercizio 3

4.3.1 Traccia

Di seguito è illustrata l'architettura di una rete di computer in un sistema bancario. L'architettura è denominata skip-ring network ed è progettata per consentire ai processori di comunicare anche dopo che si sono verificati guasti ai nodi. Ad esempio, se il nodo 1 si guasta, il nodo 8 può bypassare il nodo guasto instradando i dati sul collegamento alternativo che collega i nodi 8 e 2. Supponendo che i collegamenti siano perfetti e che i nodi abbiano ciascuno una reliability di R_m , derivare un'espressione per la reliability di tale rete. Se R_m obbedisce alla legge di failure esponenziale e il tasso di failure di ciascun nodo è 0,005 guasti all'ora, determinare la reliability del sistema alla fine di un periodo di 48 ore.

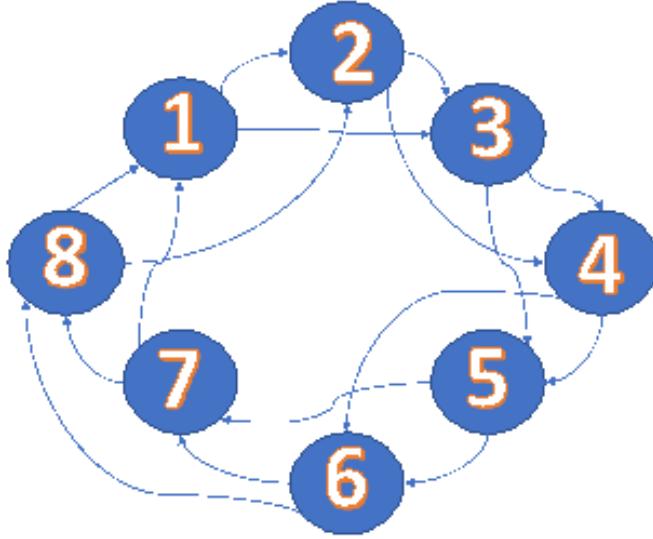


Figura 4.10: Traccia esercizio 3

4.3.2 Risoluzione

Per il corretto funzionamento della rete skip-ring è necessario che si verifichi una precisa condizione: non devono fallire due nodi adiacenti. Questo perché, se questa condizione si verificasse, la rete non sarebbe più in grado di instradare i dati su un collegamento alternativo.

Per calcolare la reliability di questo sistema, possiamo partire dalla formula per i sistemi *M-out-of-N*, che sono sistemi composti da N componenti, ciascuno con reliability R_m , dei quali almeno M devono essere funzionanti:

$$R_{NM} = \sum_{i=0}^{N-M} \binom{N}{i} \cdot R_m^{N-i} \cdot (1 - R_m)^i$$

Il problema in esame è di tipo M-out-of-N, ma la precedente formula va leggermente modificata andando a sottrarre al coefficiente binomiale tutte le combinazioni in cui il sistema non funziona.

Sono stati quindi valutati tutti i fattori che contribuiscono alla reliability totale:

- $k = 0$: Se tutti i nodi funzionano correttamente, il sistema funziona sempre, e sostituendo nella formula si ottiene:

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

- $k = 1$: Se fallisce un solo nodo, il sistema continua a funzionare sempre, e sostituendo nella formula si ottiene:

$$\binom{8}{1} R_m^7 (1 - R_m) = 8 \cdot R_m^7 (1 - R_m)$$

- $k = 2$: Se falliscono due nodi, vi sono 8 combinazioni per le quali il sistema non funziona, che si verificano quando a fallire sono due nodi adiacenti. Otteniamo quindi:

$$\left[\binom{8}{2} - 8 \right] R_m^6 (1 - R_m)^2 = 20 \cdot R_m^6 (1 - R_m)^2$$

- $k = 3$: Se falliscono tre nodi, vi sono due casi in cui il sistema non funziona:

- Falliscono 3 nodi adiacenti: 8 combinazioni possibili
- Falliscono 2 nodi adiacenti e un terzo non adiacente: per ogni coppia di nodi adiacente che fallisce vi sono 4 nodi non adiacenti che possono fallire, e poiché le possibili combinazioni di coppie di nodi adiacenti sono 8 otteniamo $8 \cdot 4 = 32$ possibili combinazioni di questo tipo.

Vi sono quindi $32 + 8 = 40$ combinazioni in cui il sistema fallisce che vanno sottratte al coefficiente binomiale, ottenendo:

$$\left[\binom{8}{3} - 40 \right] R_m^5 (1 - R_m)^3 = 16 \cdot R_m^5 (1 - R_m)^3$$

- $k = 4$: Se falliscono 4 nodi è possibile ragionare in maniera duale, valutando quali combinazioni permettono al sistema di continuare a funzionare. Queste sono quelle in cui i nodi falliti si alternano con quelli funzionanti, ossia:

- 1-3-5-7 funzionano, 2-4-6-8 falliscono
- 2-4-6-8 funzionano, 1-3-5-7 falliscono

Vi sono quindi soltanto 2 combinazioni in cui il sistema funziona, e possiamo scrivere la reliability come:

$$2 \cdot R_m^4 (1 - R_m)^4$$

- $5 \leq k \leq 8$: non è presente nessuna combinazione che permette al sistema di funzionare correttamente.

A questo punto, supponendo che la reliability del singolo componente segua la legge di fallimento esponenziale con un failure rate per il singolo nodo pari a $\lambda = 0.005$ failure/h, è possibile riscrivere R_m nel seguente modo:

$$R_m = e^{-\lambda t} = e^{-0.005t}$$

Possiamo quindi ottenere la reliability del sistema complessivo sommando i vari contributi ottenuti al variare di k :

$$R_{sys}(t) = R_m^8 + 8 \cdot R_m^7 (1 - R_m) + 20 \cdot R_m^6 (1 - R_m)^2 + 16 R_m^5 (1 - R_m)^3 + 2 R_m^4 (1 - R_m)^4$$

Valutandola dopo un periodo di 48 ore ($t = 48$), otteniamo il valore della reliability del sistema:

$$R_{sys}(48h) \approx 0.73$$

Un altro modo per studiare questa rete è effettuare una **fault trees analysis**. I fault tree possono essere visti come il duale degli RBD: mentre negli RBD l'obiettivo era comprendere le condizioni in cui il sistema funziona, nei fault trees ci si focalizza sulle condizioni che portano al fallimento del sistema. I componenti del sistema sono rappresentati come nodi dell'albero, i quali possono assumere un valore booleano a seconda che il componente corrispondente sia fallito (1) o meno (0). In particolare, si avrà il fallimento del sistema complessivo se l'uscita del nodo più alto dell'albero sarà pari a 1. Poichè i *basic event*, cioè i fallimenti dei singoli componenti, possono essere combinati utilizzando operatori logici, il modello diventa una rete combinatoria logica.

Di seguito è riportato il fault tree in logica positiva per la rete in esame:

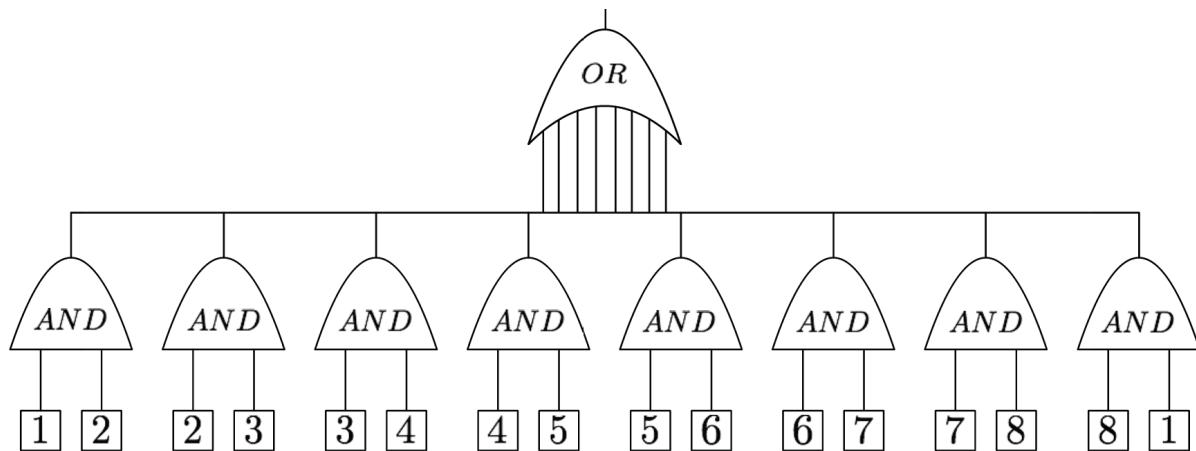


Figura 4.11: Fault Tree

Si noti come il fault tree riportato presenta alcuni eventi ripetuti, ossia componenti contenuti in diverse foglie dell'albero; da ciò ne consegue che i success path del problema non siano in realtà indipendenti. Se si intedesse calcolare la reliability del sistema a partire dal fault tree stesso, si renderebbe necessario l'utilizzo del conditioning.

4.4 Esercizio 4

4.4.1 Traccia

Comparare la reliability dei seguenti schemi, assumendo una occorrenza di failure esponenziale con i seguenti valori:

- $MTTF_A = 1000h$
- $MTTF_B = 5000h$
- $MTTF_C = 2000h$

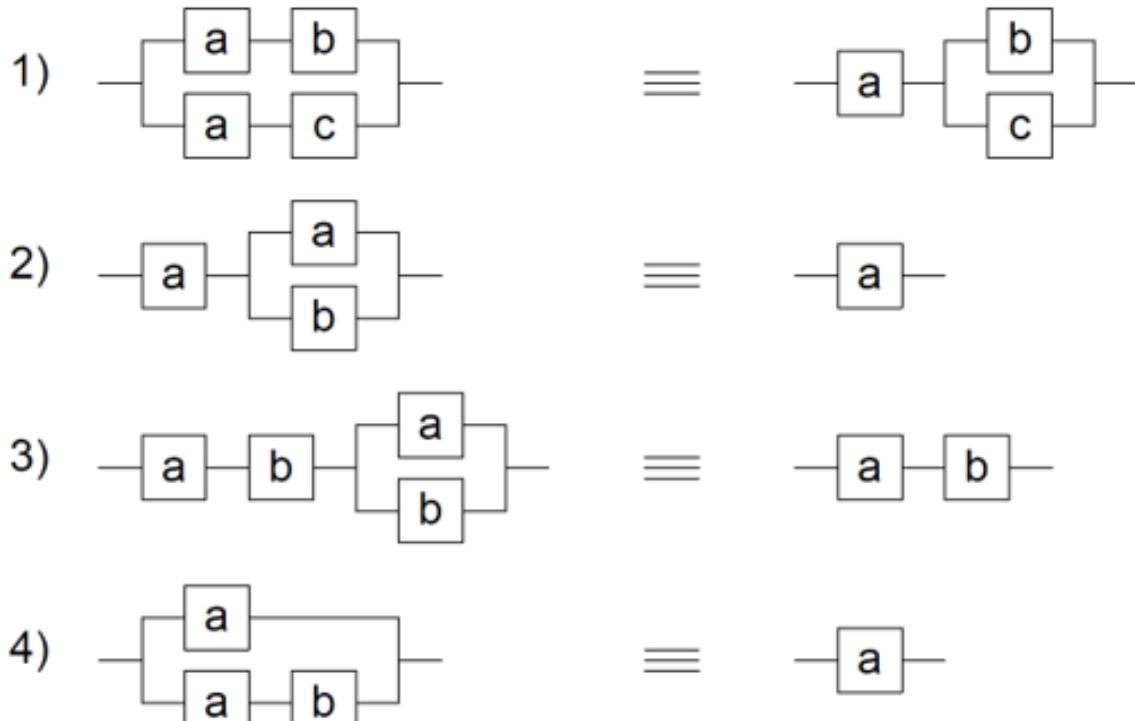


Figura 4.12: Traccia esercizio 4

Schema 1

La reliability del primo sistema può essere facilmente calcolata considerando la configurazione come parallelo di due serie, una composta da A-B e una da A-C.

$$\begin{aligned}
 R_{S1} &= 1 - (1 - R_A R_B)(1 - R_A R_C) \\
 &= 1 - \left(1 - e^{-\frac{1}{1000}t} e^{-\frac{1}{5000}t}\right) \left(1 - e^{-\frac{1}{1000}t} e^{-\frac{1}{2000}t}\right) \\
 &= 1 - \left(1 - e^{-\frac{3}{25000}t}\right) \left(1 - e^{-\frac{3}{2000}t}\right) = e^{-\frac{3}{2000}t} + e^{-\frac{3}{2500}t} - e^{-\frac{27}{10000}t}
 \end{aligned}$$

Per quanto riguarda il secondo sistema abbiamo invece una serie di due componenti, dove il primo è semplicemente A mentre il secondo è un parallelo tra B-C.

$$\begin{aligned}
 R_{S2} &= R_A (1 - (1 - R_B) (1 - R_C)) \\
 &= e^{-\frac{1}{1000}t} \left(1 - \left(1 - e^{-\frac{1}{5000}t} \right) \left(1 - e^{-\frac{1}{2000}t} \right) \right) \\
 &= e^{-\frac{1}{1000}t} \left(e^{-\frac{1}{2000}t} + e^{-\frac{1}{5000}t} - e^{-\frac{7}{10000}t} \right) = e^{-\frac{3}{2000}t} + e^{-\frac{3}{2500}t} - e^{-\frac{17}{10000}t}
 \end{aligned}$$

Tracciamo ora il grafico al variare del tempo per confrontare la reliability delle due configurazioni:

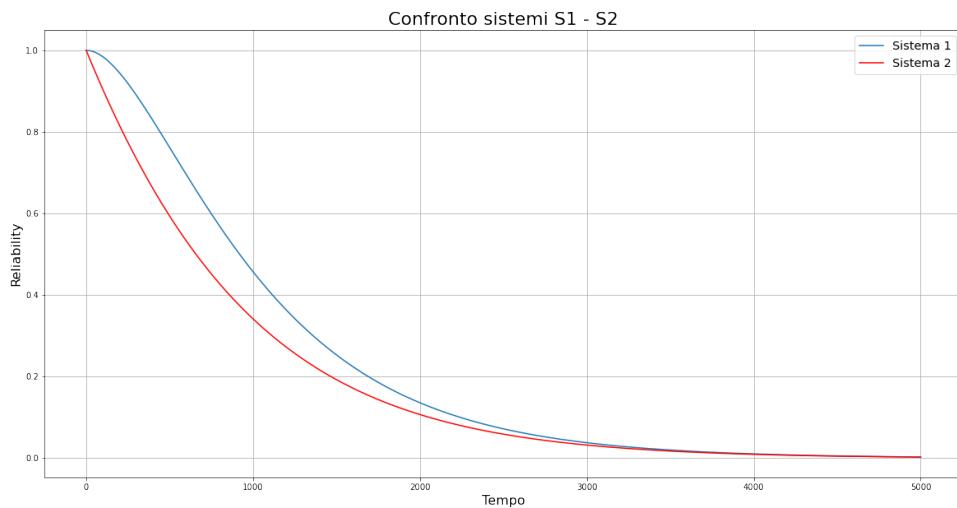


Figura 4.13: Confronto sistemi S1 - S2

Notiamo come la curva del sistema 1 (in blu) sia sempre al di sopra della curva del sistema 2 (in rosso), quindi possiamo concludere che lo schema 1 sia il migliore.

Schema 2

La reliability del primo sistema può essere calcolata considerando lo schema come una serie di due componenti, dove uno è semplicemente A e l'altro è il parallelo tra A-B.

$$\begin{aligned}
 R_{S1} &= R_A (1 - (1 - R_A) (1 - R_B)) \\
 &= e^{-\frac{1}{1000}t} \left(1 - \left(1 - e^{-\frac{1}{1000}t} \right) \left(1 - e^{-\frac{1}{500}t} \right) \right) \\
 &= e^{-\frac{1}{1000}t} \left(-e^{-\frac{3}{2500}t} + e^{-\frac{1}{500}t} + e^{-\frac{1}{1000}t} \right) = e^{-\frac{3}{2500}t} + e^{-\frac{1}{500}t} - e^{-\frac{11}{5000}t}
 \end{aligned}$$

Il secondo sistema, invece, è semplicemente A.

$$R_{S2} = R_A = e^{-\frac{1}{1000}t}$$

Tracciamo ora il grafico al variare del tempo per confrontare la reliability delle due configurazioni:

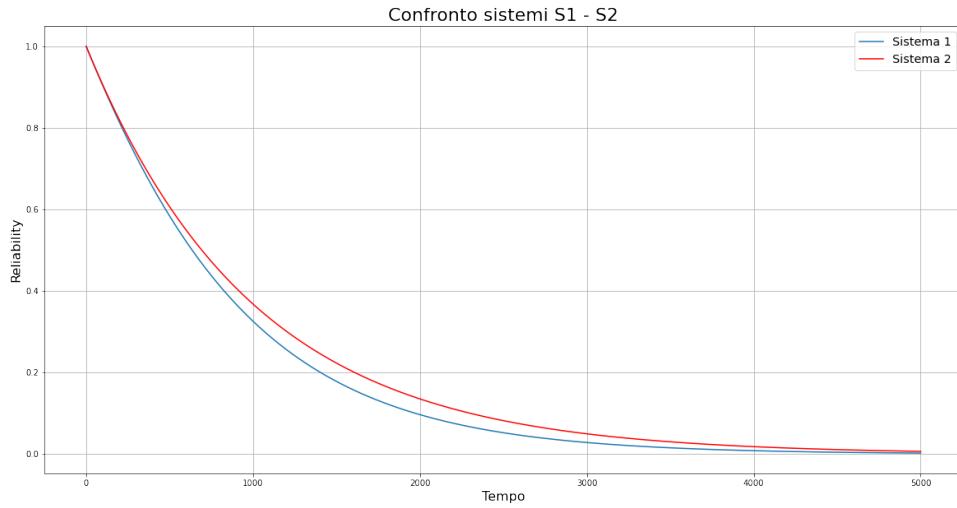


Figura 4.14: Confronto sistemi S1 - S2

Notiamo come la curva del sistema 2 (in rosso) sia sempre al di sopra della curva del sistema 1 (in blu), quindi possiamo concludere che lo schema 2 sia il migliore.

Schema 3

La reliability del primo sistema può essere calcolata considerando lo schema come una serie di tre componenti: il primo è semplicemente A, il secondo semplicemente B, mentre il terzo è un parallelo tra A-B.

$$\begin{aligned}
 R_{S1} &= R_A R_B (1 - (1 - R_A)(1 - R_B)) \\
 &= e^{-\frac{1}{1000}t} e^{-\frac{1}{5000}t} \left(1 - \left(1 - e^{-\frac{1}{1000}t}\right) \left(1 - e^{-\frac{1}{5000}t}\right)\right) \\
 &= e^{-\frac{3}{2500}t} \left(e^{-\frac{1}{5000}t} + e^{-\frac{1}{1000}t} - e^{-\frac{3}{2500}t}\right) \\
 &= e^{-\frac{7}{5000}t} + e^{-\frac{11}{5000}t} - e^{-\frac{6}{2500}t}
 \end{aligned}$$

Il secondo sistema, invece, è semplicemente la serie tra A-B.

$$R_{S2} = R_A R_B = e^{-\frac{1}{1000}t} e^{-\frac{1}{5000}t} = e^{-\frac{3}{2500}t}$$

Tracciamo ora il grafico al variare del tempo per confrontare la reliability delle due configurazioni:

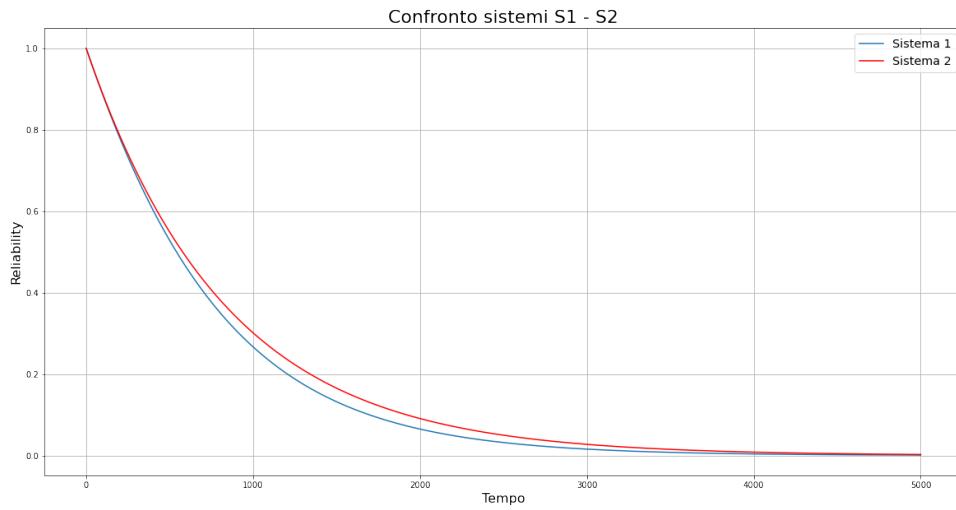


Figura 4.15: Confronto sistemi S1 - S2

Notiamo come la curva del sistema 2 (in rosso) sia sempre al di sopra della curva del sistema 1 (in blu), quindi possiamo concludere che lo schema 2 sia il migliore.

Schema 4

La reliability del primo sistema può essere calcolata considerando lo schema come un parallelo di due componenti: il primo è semplicemente A, mentre il secondo è la serie di A-B.

$$\begin{aligned}
 R_{S1} &= 1 - (1 - R_A)(1 - R_A R_B) \\
 &= 1 - \left(1 - e^{-\frac{1}{1000}t}\right) \left(1 - e^{-\frac{1}{1000}t} e^{-\frac{1}{5000}t}\right) \\
 &= e^{-\frac{3}{2500}t} + e^{-\frac{1}{1000}t} - e^{-\frac{11}{5000}t}
 \end{aligned}$$

Il secondo sistema è semplicemente A.

$$R_{S2} = R_A = e^{-\frac{1}{1000}t}$$

Tracciamo ora il grafico al variare del tempo per confrontare la reliability delle due configurazioni:

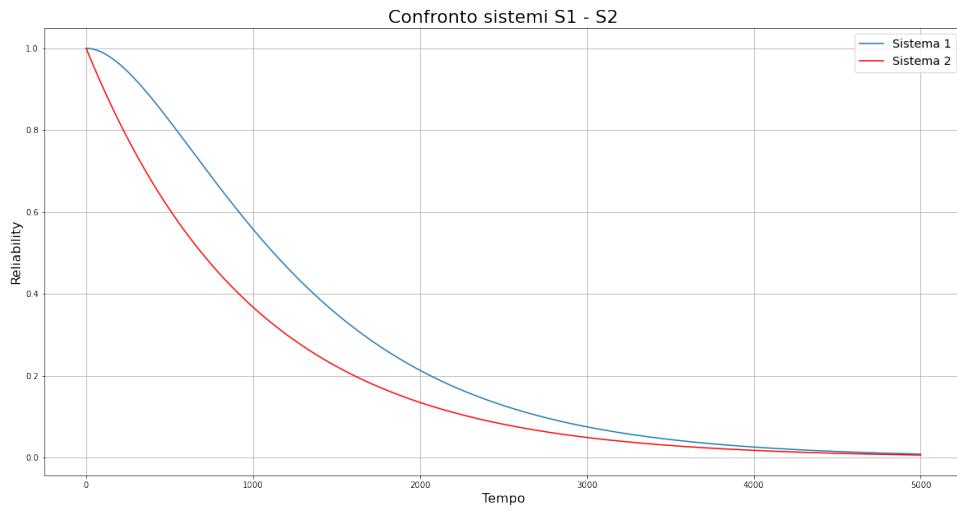


Figura 4.16: Confronto sistemi S1 - S2

Notiamo come la curva del sistema 1 (**in blu**) sia sempre al di sopra della curva del sistema 2 (**in rosso**), quindi possiamo concludere che lo schema 1 sia il migliore.

4.5 Esercizio 5

4.5.1 Traccia

Il sistema mostrato nella figura seguente è un sistema di elaborazione per un elicottero. Il sistema dispone di processori a doppia ridondanza e unità di interfaccia a doppia ridondanza. Nel sistema vengono utilizzati due bus e ogni bus è anch'esso a doppia ridondanza. La parte interessante del sistema è l'apparecchiatura di navigazione. Il velivolo può essere pilotato completamente utilizzando il sistema di navigazione inerziale (INS). Se l'INS fallisce, il velivolo può essere pilotato utilizzando la combinazione del Doppler e della rotta di altitudine e del sistema di riferimento (AHRS). Il sistema contiene tre unità AHRS, di cui solo una è necessaria. Questo è un esempio di ridondanza funzionale in cui i dati dell'AHRS e del Doppler possono essere utilizzati per sostituire l'INS, se l'INS fallisce. A causa degli altri sensori e strumentazione, entrambi i bus sono necessari per il corretto funzionamento del sistema indipendentemente dalla modalità di navigazione utilizzata.

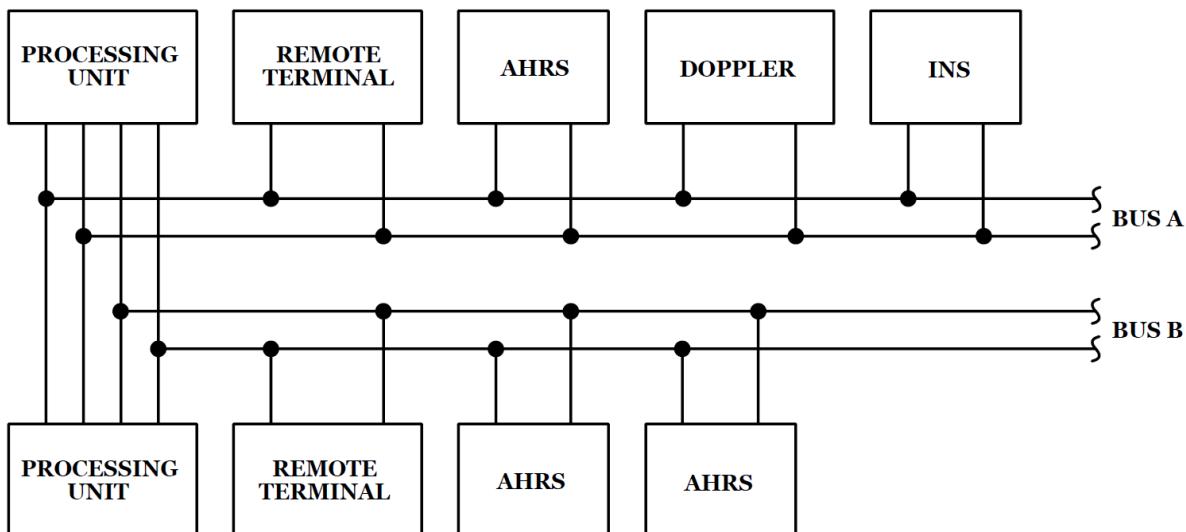


Figura 4.17: Traccia esercizio 5

- Disegnare il reliability block diagram del sistema.
- Disegnare il fault tree del sistema e analizzare i cut set.
- Calcolare la reliability per un volo di un'ora utilizzando i dati MTTF riportati nella tabella sottostante. Si suppone che si applichi la legge di fallimento esponenziale e che la fault coverage sia perfetta.

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

Tabella 4.1: MTTF per ogni componente

- (d) Ripetere (c), ma questa volta incorporare un fattore di coverage per la fault detection e la riconfigurazione delle processing unit. Utilizzando gli stessi dati di MTTF, determinare il valore approssimativo di fault coverage necessario per ottenere una reliability (alla fine di un'ora) di 0.99999.

4.5.2 Risoluzione (a)

L'RBD del sistema può essere visto come la serie di diversi blocchi:

- BUS A: Il bus A è dual redundant, quindi può essere rappresentato come parallelo di due bus A.
- BUS B: Il bus B è dual redundant, quindi può essere rappresentato come parallelo di due bus B.
- CPU: La processing unit è dual redundant, quindi può essere rappresentata come parallelo di due PU.
- RT: Il remote terminal è dual redundant, quindi può essere rappresentato come parallelo di due RT.
- Navigation equipment: Il blocco per la navigazione è composto dal parallelo di due sottoblocchi, di cui il primo è semplicemente l'INS, mentre il secondo è composto dalla serie tra DOPPL (doppler) e il parallelo di 3 AHRS.

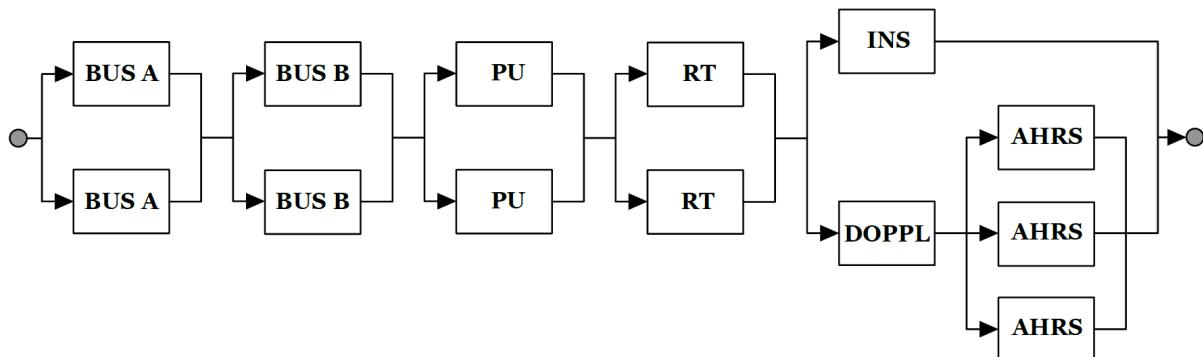


Figura 4.18: Reliability Block Diagram

4.5.3 Risoluzione (b)

E' possibile generare un fault tree a partire dall'RBD appena tracciato operando le seguenti trasformazioni:

- I blocchi connessi in serie saranno ingressi di porte OR: il fallimento di uno di essi comporta il fallimento del (sotto)sistema.
- I blocchi connessi in parallelo saranno ingressi di porte AND: il fallimento del (sotto)sistema deriva dal fallimento di tutti i blocchi che lo compongono.

Otteniamo quindi il seguente fault tree:

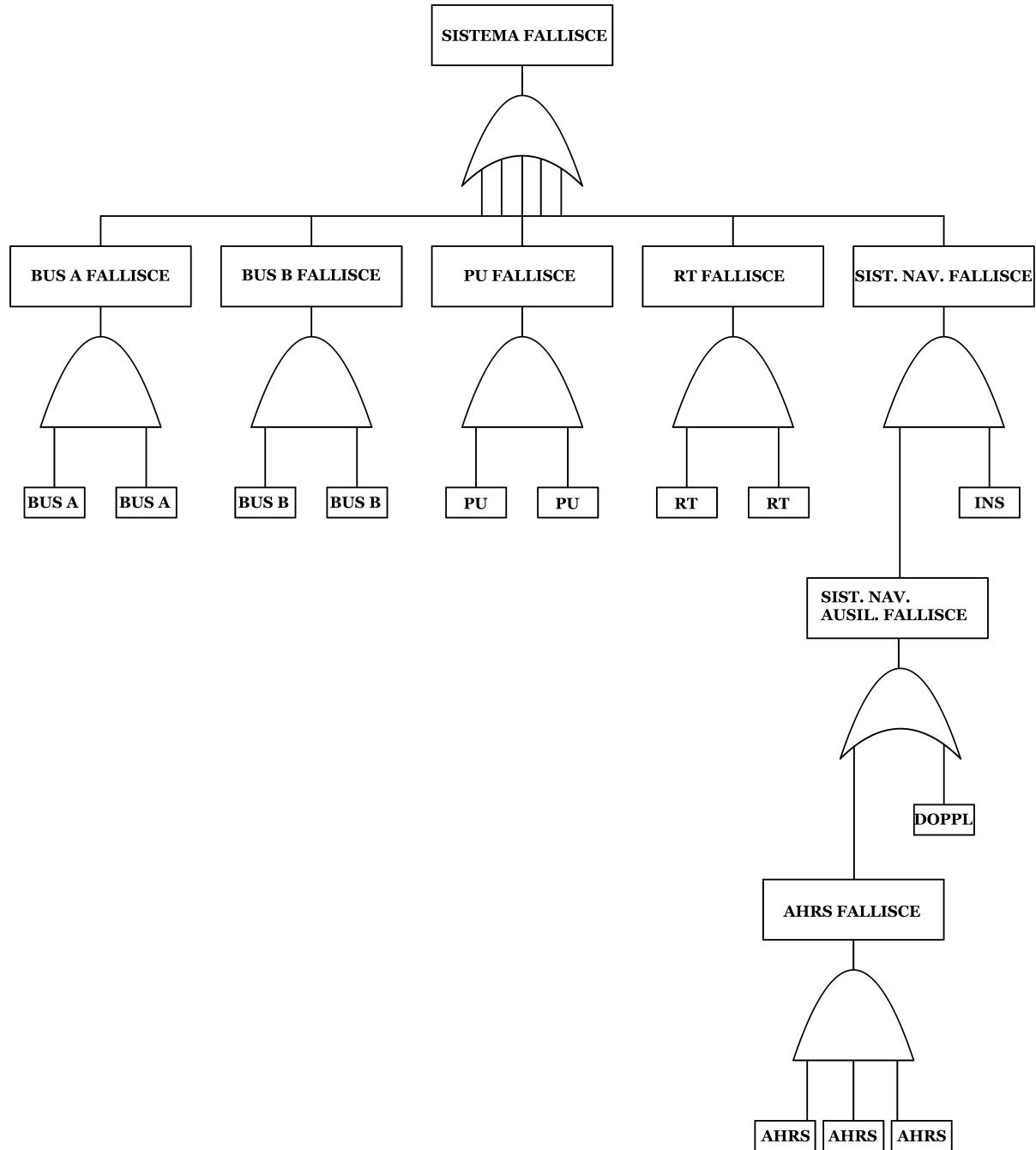


Figura 4.19: Fault Tree

A questo punto si può procedere con la fault tree analysis, che comporta l'individuazione di tutte le possibili combinazioni di basic faults che determinano il fallimento del sistema complessivo. Quindi, il fallimento del sistema può essere descritto mediante la seguente espressione logica, ottenuta traducendo il fault tree nella prima forma canonica (somma di prodotti):

$$\begin{aligned}
 TLE &= (BUSA_1 \cdot BUSA_2) + (BUSB_1 \cdot BUSB_2) + (PU_1 \cdot PU_2) + (RT_1 \cdot RT_2) + \\
 &\quad + (INS \cdot (DOPPL + (AHRS_1 \cdot AHRS_2 \cdot AHRS_3))) \\
 &= (BUSA_1 \cdot BUSA_2) + (BUSB_1 \cdot BUSB_2) + (PU_1 \cdot PU_2) + (RT_1 \cdot RT_2) + \\
 &\quad + (INS \cdot DOPPL) + (INS \cdot AHRS_1 \cdot AHRS_2 \cdot AHRS_3)
 \end{aligned}$$

Si ottengono quindi i seguenti **cutset**, tra cui i primi 5 rappresentano i **minimal cutset** del sistema:

- $BUSA_1 \cdot BUSA_2$
- $BUSB_1 \cdot BUSB_2$
- $PU_1 \cdot PU_2$
- $RT_1 \cdot RT_2$
- $INS \cdot DOPPL$
- $INS \cdot AHRS_1 \cdot AHRS_2 \cdot AHRS_3$

4.5.4 Risoluzione (c)

A questo punto è possibile calcolare la reliability per un volo di un'ora utilizzando i valori per MTTF dati, supponendo di applicare la legge di fallimento esponenziale e di avere una fault coverage perfetta.

Per farlo si ricava la seguente formula a partire dal fault tree (o analogamente dall'RBD):

$$\begin{aligned}
 R_{sys}(t) &= (1 - (1 - R_{BUS})^2) (1 - (1 - R_{BUS})^2) (1 - (1 - R_{PU})^2) (1 - (1 - R_{RT})^2) \\
 &\quad \cdot (1 - (1 - R_{INS}) (1 - R_{AHRS})^3) (1 - (1 - R_{INS}) (1 - R_{DOPPL})) = \\
 &= \left(1 - \left(1 - e^{-\frac{1}{60000}t}\right)^2\right)^2 \left(1 - \left(1 - e^{-\frac{1}{10000}t}\right)^2\right) \left(1 - \left(1 - e^{-\frac{1}{4500}t}\right)^2\right) \\
 &\quad \cdot \left(1 - \left(1 - e^{-\frac{1}{2000}t}\right) \left(1 - e^{-\frac{1}{2000}t}\right)^3\right) \left(1 - \left(1 - e^{-\frac{1}{2000}t}\right) \left(1 - e^{-\frac{1}{500}t}\right)\right)
 \end{aligned}$$

Sostituendo $t = 1$ (volo di un'ora) otteniamo:

$$R_{sys}(t) \approx 0.9999989413$$

4.5.5 Risoluzione (d)

Infine è richiesto di ripetere i calcoli del punto (c) incorporando un fattore di coverage per la fault detection e la riconfigurazione delle processing unit.

La fault coverage si riferisce alla percentuale di alcuni tipi di guasti che possono essere rilevati durante il test di qualsiasi sistema ingegnerizzato. L'effetto di coverage è dato da un fattore c che è la probabilità che il componente che fallisce riconosca il proprio fallimento e riesca a riconfigurarsi, e quindi a non portare al fallimento dell'intero sistema.

In questo sistema è necessario introdurre un fattore di coverage per quanto riguarda le processing unit (PU), che sono collegate tra di loro in parallelo. Nel caso di due sistemi in parallelo nei quali la failure detection non è perfetta si introduce il fattore di coverage c e si la seguente formula per la reliability del sistema:

$$R_{sys} = R_1 + c \cdot (1 - R_1) \cdot R_2$$

Particularizzando al caso in esame, la reliability del parallelo tra le due PU diventa:

$$R_{PUs} = R_{PU} + c \cdot (1 - R_{PU}) \cdot R_{PU}$$

Sostituendo nella formula della reliability totale del sistema, otteniamo:

$$\begin{aligned} R_{sys}(t) &= (1 - (1 - R_{BUS})^2) (1 - (1 - R_{BUS})^2) (R_{PU} + c \cdot (1 - R_{PU}) \cdot R_{PU}) \\ &\quad \cdot (1 - (1 - R_{RT})^2) (1 - (1 - R_{INS}) (1 - R_{AHRS})^3) (1 - (1 - R_{INS}) (1 - R_{DOPPL})) \end{aligned}$$

L'obiettivo finale è determinare il valore approssimativo di fault coverage necessario per ottenere una reliability dopo un'ora di volo di *five nines* (0.99999). Per fare ciò si pone $R_{sys}(t) = 0.99999$ con $t = 1$ e si risolve rispetto a c l'equazione della reliability totale, ottenendo:

$$c \geq 0.910574$$

Es. 5 - FFDA

5.1 Traccia

Effettuare una FFDA (Field Failure Data Analysis) basata su log (già filtrati) dei seguenti supercalcolatori:

- **Mercury**, appartenente al *National Center for Supercomputing Application* (NCSA at University of Illinois).
- **BlueGene/L**, del *Lawrence Livermore National Labs* (LLNL, CA, USA).

Per l'analisi, sono proposti i seguenti quesiti:

- Mercury e BlueGene/L: selezionare preventivamente i primi 5 nodi con più entry; determinare CWIN e il conteggio delle tuple per ciascun nodo:
 1. CWIN è lo stesso per tutti i nodi?
 2. esistono bottleneck di reliability, cioè nodi con un gran numero di tuple rispetto agli altri?
 3. tracciare la reliability a livello di nodo per i nodi con numero di entry significativamente grande.
- Mercury: per ogni categoria di errore (escluso OTH) determinare CWIN, il conteggio delle tuple e il modello di reliability:
 1. qual è la categoria con il maggior numero di tuple?
 2. quale categoria è meno / più reliable?
- BlueGene/L: analizza le categorie J18-U01 e J18-U11 (tupling e reliability).
- Mercury e BlueGene/L: concentrarsi su nodi funzionali simili (ad es. $tg-c$ in Mercury, 2 nodi di I-O in BG/L, ecc.) selezionati tra i nodi con più entry a riguardo: i nodi mostrano conteggi di tuple / parametri di reliability simili?
- Mercury: estrarre i tipi di errore e i nodi che danno più contributo a tali tipologie di errore. Cosa si può notare?
- BlueGene/L: quali sono i rack / nodi con più entry?

5.2 Strumenti utilizzati

L'analisi di tale esercizio è stata affrontata utilizzando notebook *Python* sviluppati nell'ambiente Jupyter Notebook.

5.3 Introduzione - FFDA

La Field Failure Data Analysis consiste nell'analisi di dati riguardanti fallimenti di macchine o più in generale di impianti di elaborazione, raccolti sul campo. Essendo un'analisi effettuata a partire da workload reali a cui è sottoposto il sistema stesso (e che ha portato a dei fallimenti), la FFDA permette di effettuare una misurazione diretta alla reliability del sistema esaminato. Tale tecnica prevede in genere il susseguirsi delle seguenti fasi:

- **Data logging e data collection:** fase di raccolta dei dati, in cui è necessario chiarire quali dati raccogliere e come raccoglierli. A tal fine, si rende necessario uno studio preliminare del sistema per individuare le migliori metodologie per realizzare questa fase.
- **Data filtering:** filtraggio dei dati grezzi, raccolti allo step precedente. Lo scopo di questo passo è quello di ridurre la quantità di informazioni in gioco, in modo da focalizzare l'attenzione esclusivamente sui dati ritenuti significativi.
- **Data manipulation:** estrazione di errori e/o fallimenti nel sistema a partire dai dati filtrati. Nel dettaglio, si è scelto di utilizzare tecniche di coalescenza per condurre tale fase di sviluppo, le quali cercano di individuare una certa tipologia di correlazione tra i dati.
- **Data analysis:** analisi vera e propria dei dati ottenuti dalle fasi precedenti. Nel dettaglio, l'analisi è realizzata attraverso delle valutazioni statistiche sui dati, al fine di vagliarne alcune metriche quantitative.

5.4 Mercury

Il supercalcolatore Mercury presenta diversi nodi di elaborazione IBM, organizzati in un'architettura 3-layer:

- **login nodes**
- **computational nodes**
- **storage nodes**

A tali nodi si aggiunge un nodo di management (**master node**).

L'architettura prevede inoltre l'organizzazione di tali nodi in differenti sottosistemi, quali *Processor, Device, Memory, Network, I-O, Other*.

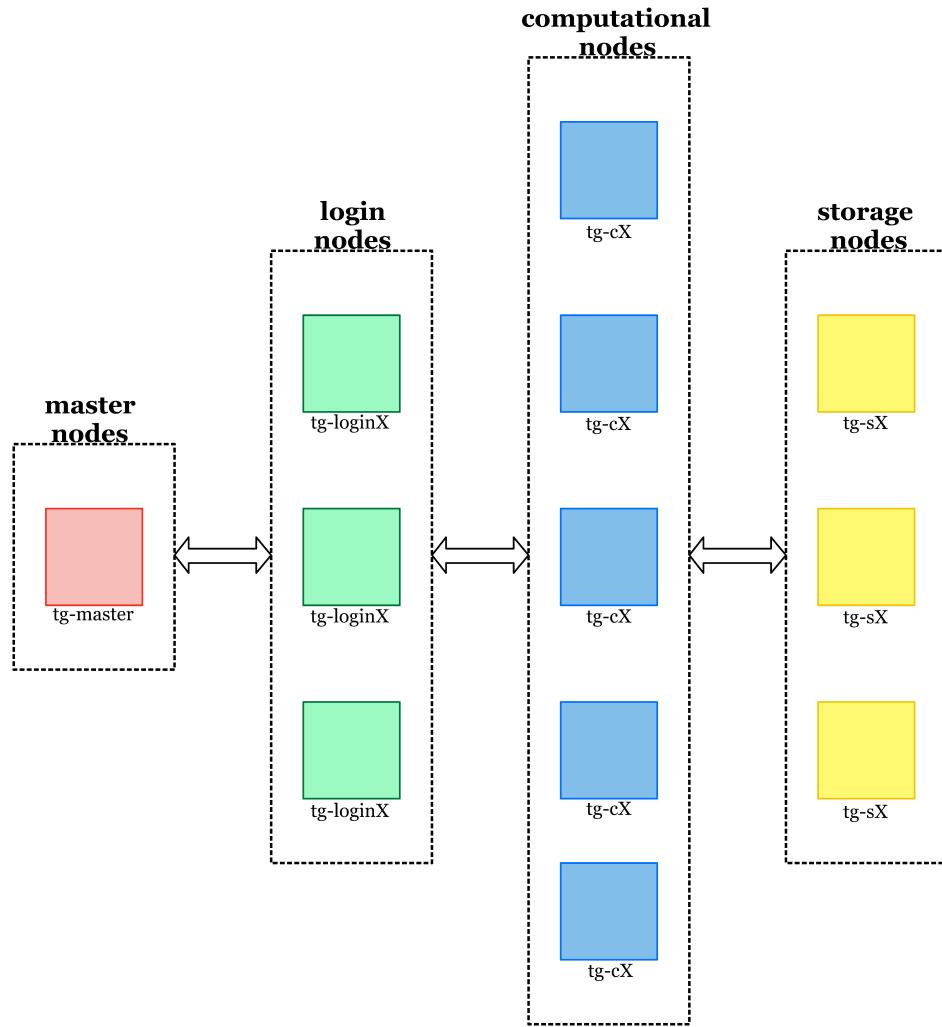


Figura 5.1: Schema architetturale - Mercury

5.4.1 Log Analysis

Essendo dinanzi a file di log già collezionati e opportunamente filtrati, la FFDA in tal caso verrà condotta a partire dalla fase di data manipulation. Tuttavia, è necessario analizzare preventivamente i log filtrati, in modo da comprendere a pieno il problema in esame.

I log di Mercury sono composti dalle seguenti informazioni:

- *Timestamp*
- *Originating node*: nodo che ha generato l'errore.
- *Originating subsystem*: sottosistema in errore, che rappresenta la tipologia di errore verificatasi. Le tipologie di errore sono: PRO, DEV, MEM, NET, I-O, OTH.
- *Message*: stringa riportante il messaggio di errore esteso.

	Timestamp	Node	Subsystem	Message
0	1167637660	tg-c645	PRO	+BEGIN HARDWARE ERROR STATE AT CMC
1	1167637660	tg-c645	PRO	Device Error Info Section
2	1167637660	tg-c645	PRO	Error Map: x
3	1167637720	tg-c645	PRO	+BEGIN HARDWARE ERROR STATE AT CMC
4	1167637720	tg-c645	PRO	Device Error Info Section

Figura 5.2: Mercury log

5.4.2 Data manipulation

Come detto in precedenza, la fase di data manipulation si occupa dell'estrazione di eventi, corrispondenti a specifici fallimenti del sistema, dai dati filtrati. La tecnica di **coalescenza** permette di individuare una correlazione tra i dati in gioco; nel dettaglio, si è scelto di utilizzare una **coalescenza spaziale**, in modo tale da individuare una correlazione temporale tra le entry del log, le quali sono però appartenenti a nodi differenti del sistema. Tale tecnica ci permette di vagliare come i fallimenti si propaghino tra i diversi nodi del sistema Mercury.

I risultati prodotti dalla tecnica di coalescenza temporale sono delle **tuple**, ossia un insieme di eventi correlati tra loro, i quali potrebbero essere probabilmente associati allo stesso fallimento.

L'algoritmo che permette di realizzare il **tupling** può essere schematizzato come segue:

```

if (ts( $x_{i+1}$ ) – ts( $x_i$ ) < W) then
    aggiungi  $x_{i+1}$  alla tupla corrente
else
    genera una nuova tupla contenente  $x_{i+1}$ 

```

Un primo problema consiste nel corretto dimensionamento della finestra di coalescenza W , poiché:

- una finestra temporale troppo grande potrebbe portare a dei collassi, raggruppando eventi relativi a fallimenti differenti nella stessa tupla
- una finestra temporale troppo piccola potrebbe causare dei troncamenti, scindendo eventi appartenenti allo stesso guasto in tuple differenti.

Per tale motivo si rende necessaria una **sensitivity analysis**, consistente nel ripetere il tupling per diversi valori di W , riportando il numero di tuple in relazione alla finestra stessa.

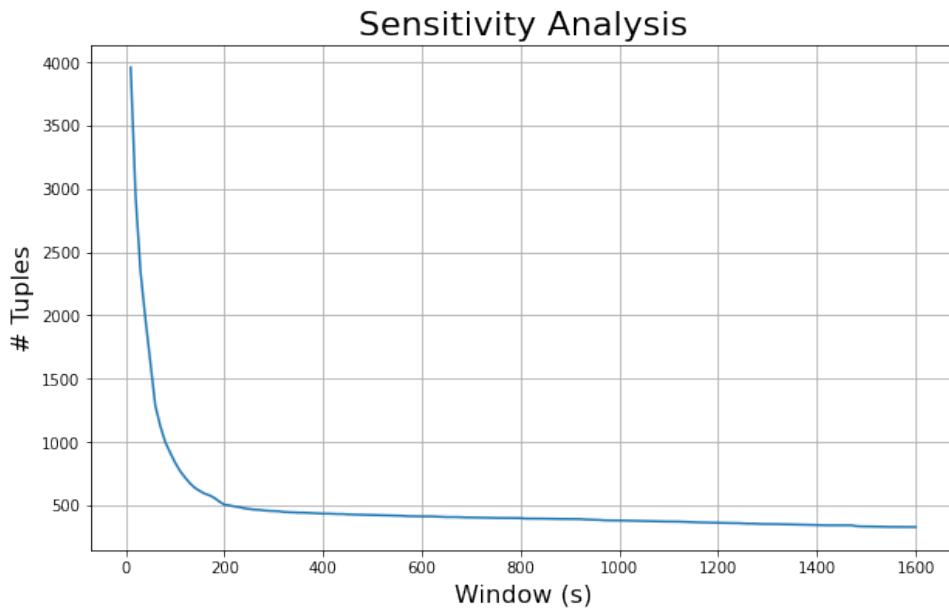


Figura 5.3: Sensitivity Analysis

Un buon compromesso per il valore della finestra temporale W è in genere quello appena successivo al gomito della curva, in quanto la zona a discesa rapida è sintomatica di molteplici troncamenti, mentre al contrario una discesa dolce in genere rispecchia dei collassi. Per tale motivo si è scelto una finestra temporale **$W = 200$** .

Tramite tale valore della finestra di coalescenza, si è proceduto ad effettuare il tupling, il quale restituisce **503 tuple**. Per le varie tuple si è poi proceduto a calcolare alcune misurazioni statistiche:

- Numero di entry per tupla
- Starting point: timestamp della prima entry della tupla
- Ending point: timestamp dell'ultima entry della tupla
- Inter-arrival time: differenza temporale (in secondi) tra il timestamp della prima entry di una tupla e il timestamp dell'ultima entry della tupla precedente.

- Length: differenza tra il timestamp dell'ultima e della prima entry di ogni tupla
- Numero di nodi colpiti di fallimento, con relativo elenco di tali nodi

Tuple	Num_entries	Starting_point	Ending_point	Length	Num_nodes	Nodes	
0	1	6	1167637660	1167637720	60	1	(tg-c645,)
1	2	3	1167655228	1167655229	1	1	(tg-c238,)
2	3	61	1167657137	1167657302	165	1	(tg-c238,)
3	4	14	1167657550	1167657550	0	1	(tg-c238,)
4	5	2	1167657941	1167657941	0	1	(tg-c238,)
...	
498	499	2	1174922590	1174922608	18	1	(tg-c196,)
499	500	2	1174926975	1174926993	18	1	(tg-c196,)
500	501	4	1174929130	1174929196	66	1	(tg-master,)
501	502	562	1174942783	1174943068	285	3	(tg-c196, tg-master, tg-c128)
502	503	2	1174943925	1174943943	18	1	(tg-c196,)

	Tuples	Interarrivals
0	Tuple 1-2	17508
1	Tuple 2-3	1908
2	Tuple 3-4	248
3	Tuple 4-5	391
4	Tuple 5-6	260
...
497	Tuple 498-499	1723
498	Tuple 499-500	4367
499	Tuple 500-501	2137
500	Tuple 501-502	13587
501	Tuple 502-503	857

Figura 5.4: Statistiche delle tuple

Analisi del bottleneck

In primo luogo, tramite le statistiche appena illustrate si sono ricercati dei possibili fallimenti particolarmente critici nel sistema, confrontando tra loro il numero entry per ogni tupla.

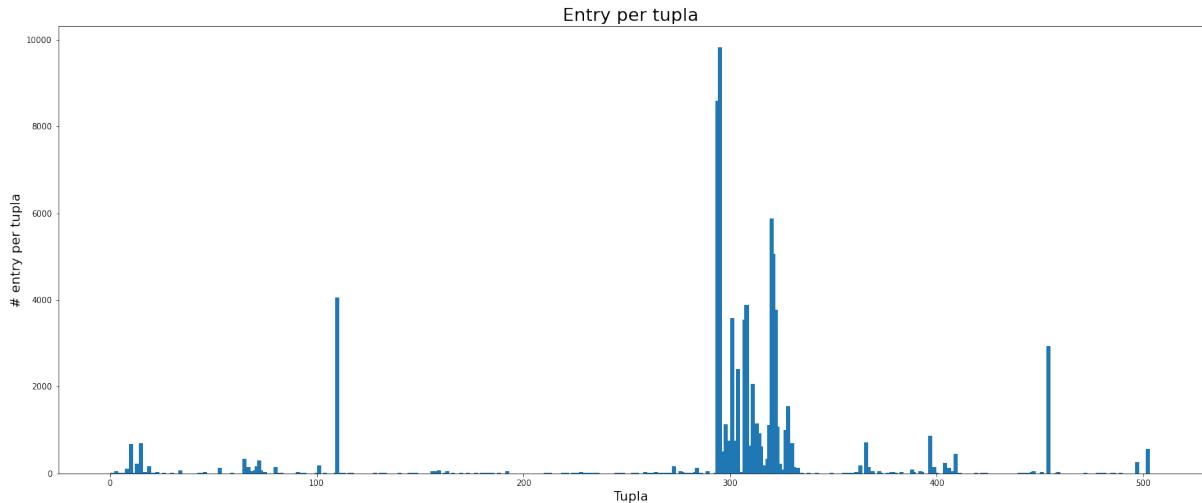


Figura 5.5: Numero di entry per ogni tupla

Da tale grafico ne conseguе come gran parte dei fallimenti complessivi del sistema siano concentrati in un insieme raccolto e contiguo di tuple; nel dettaglio, in tale insieme si registra anche la tupla con il picco massimo di entry per tupla, ossia la tupla 295 con 9825 entry, la quale copre ben il 12% dell'intero insieme di entry.

Per analizzare al meglio l'intervallo di tuple in cui si registrano la maggior parte dei fallimenti, si è realizzato il seguente grafico che evidenzia la somma cumulativa del numero di entry di ogni tupla:

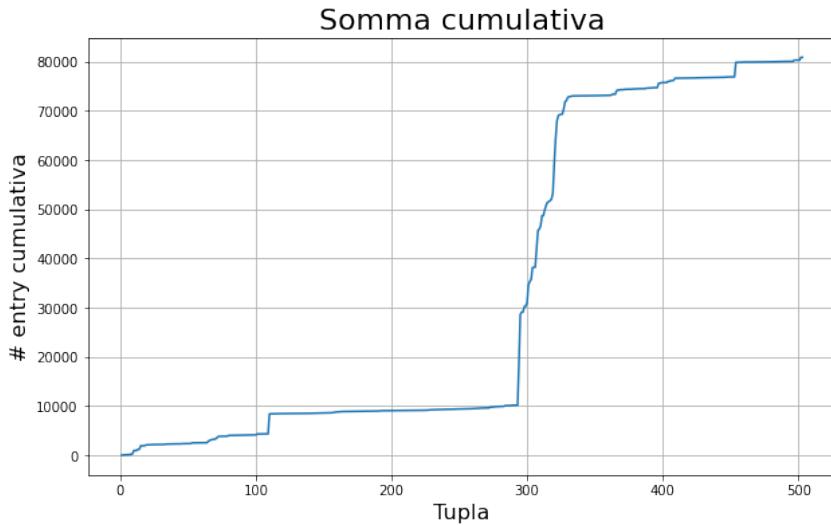


Figura 5.6: Somma cumulativa della distribuzione di entry

Si noti come la curva presenta una zona a massima pendenza nell'**intervallo di tuple 294 - 327**, con un'impennata del numero di entry da circa 10 mila fino a sfiorare le 70 mila entry. Il numero di entry contenuto in tale intervallo di tuple copre all'incirca il 75% del totale delle entry stesse.

Trasformando lo starting point della tupla 294 e l'ending point della tupla 327 in formato data, si ottiene il seguente intervallo di giorni in cui si sono registrati tali fallimenti:

- Data inizio: 2007-02-12 23:58:33
- Data fine: 2007-02-14 19:23:47

L'intervallo temporale in cui si sono verificati molti fallimenti è di **1 giorno, 19 ore, 25 minuti, 14 secondi**. Tale durata è di fatto minima se paragonata alla totalità della durata temporale coperta dai log, di ben 84 giorni.

Analizzando dettagliatamente i nodi colpiti da failure nell'intervallo di tuple appena specificato si nota come ben 60105 eventi siano stati causati dal nodo computazionale **tg-c401**, cifra che rasenta il 100% del totale di fallimenti dell'intervallo. Ciò ci permette di concludere che questo nodo è effettivamente il **bottleneck** del sistema.

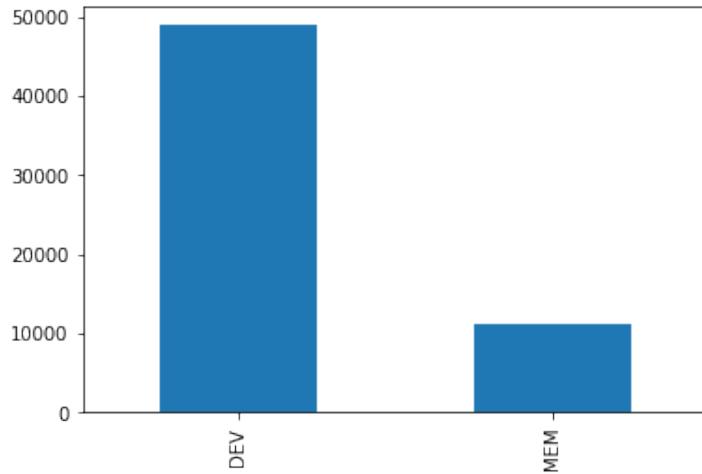


Figura 5.7: Eventi di errore per sottosistema

Si noti, inoltre, come il fallimento del nodo computazionale *tg-c401* abbia portato ad errori esclusivamente nei sottosistemi Device e Memoria. Ciò è confermato dagli messaggi di errore rilevati:

Messaggio	# occorrenze
Component Info: Vendor Id =x x	22352
Physical Address x	11152
+Platform PCI Component Error Info Section	6567
+BEGIN HARDWARE ERROR STATE AT CPE	5819
+ Platform Specific Error Detail:	5278
+Platform Specific Error Info Section	5233
+END HARDWARE ERROR STATE AT CPE	3700
+ Mem Error Detail:	4

Tabella 5.1

Analisi dei collassi

L'analisi dei collassi prevede di stimare il numero di collassi commessi nell'operazione di tupling: difatti, se la finestra temporale scelta è troppo ampia, eventi relativi a guasti diversi possono essere accorpati nella stessa tupla. Si è proceduto ad effettuare l'analisi dei collassi andando a rilevare il numero di tuple in cui sono contenuti eventi relativi a più nodi di origine. Sono 48 le tuple che presentano questa caratteristica, le quali rappresentano all'incirca il 10% delle tuple totali. Probabilmente in queste tuple potrebbero celarsi dei collassi.

Analisi dei troncamenti

L'analisi dei troncamenti consiste invece nell'individuare gli eventi dovuti allo stesso fallimento che sono stati divisi in tuple diverse, a causa di una finestra di coalescenza troppo

piccola. Tale analisi può essere condotta andando a ricercare eventi dovuti allo stesso nodo in tuple consecutive. Un'analisi di questo tipo porta ad una stima della truncation pari circa al 48%. Tale stima è piuttosto alta: una possibile causa può essere dovuta al fatto che, per come è stato costruito l'algoritmo di tupling, la distanza tra due tuple consecutive potrebbe essere piuttosto ampia. Si è quindi preventivamente individuato il valore corrispondente al *0.15-quantile* dei tempi di interarrivo, pari a 445.75 secondi, in modo da andare a considerare l'insieme di tuple consecutive il cui tempo di inter-arrivo sia inferiore al valore di quantile indicato. Solo successivamente a questa operazione si è quindi proceduto con l'effettiva verifica di tuple consecutive affette da errori relativi allo stesso nodo. Ne consegue che circa il 12% delle tuple totali potrebbe essere colpito da truncation.

5.4.3 Data Analysis

L'ultimo passo del processo di FFDA consiste nell'analisi statistica dei dati per poterne estrarre informazioni riguardanti la reliability empirica del sistema. Proprio per ricavare la stima della reliability, si è iniziato col determinare la distribuzione di probabilità(*pdf*) dei tempi di interarrivo delle tuple, rappresentanti di fatto il TTF; a partire dalla pdf empirica si è potuto poi calcolare la funzione di distribuzione cumulativa(*CDF*), che a sua volta rappresenta la probabilità che il sistema fallisca in un certo istante t ; finalmente, si può ricavare la reliability empirica del sistema come: $R(t) = 1 - CDF(t)$

Tale procedimento è stato realizzato tramite la libreria python *reliability*, la quale è in grado di realizzare in automatico anche il successivo step di **Goodness Of Fitting (GoF)**, al fine di verificare quale delle distribuzioni note è in grado di approssimare al meglio le funzioni di probabilità ottenute. Le distribuzioni note utilizzate per il fitting sono: Weibull, log-normale, Gamma, Esponenziale, log-logistica, Gumbel (tutte con diverse combinazioni di parametri).

Di seguito è riportato il risultato del procedimento:

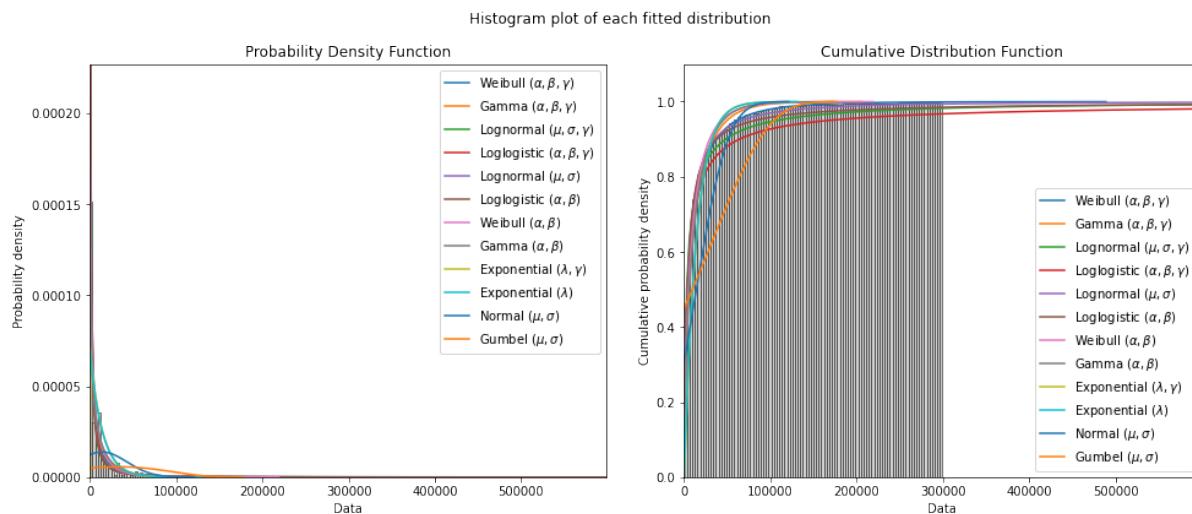


Figura 5.8: pdf & CDF empiriche, GoF

Semi-parametric Probability-Probability plots of each fitted distribution
Parametric (x-axis) vs Non-Parametric (y-axis)

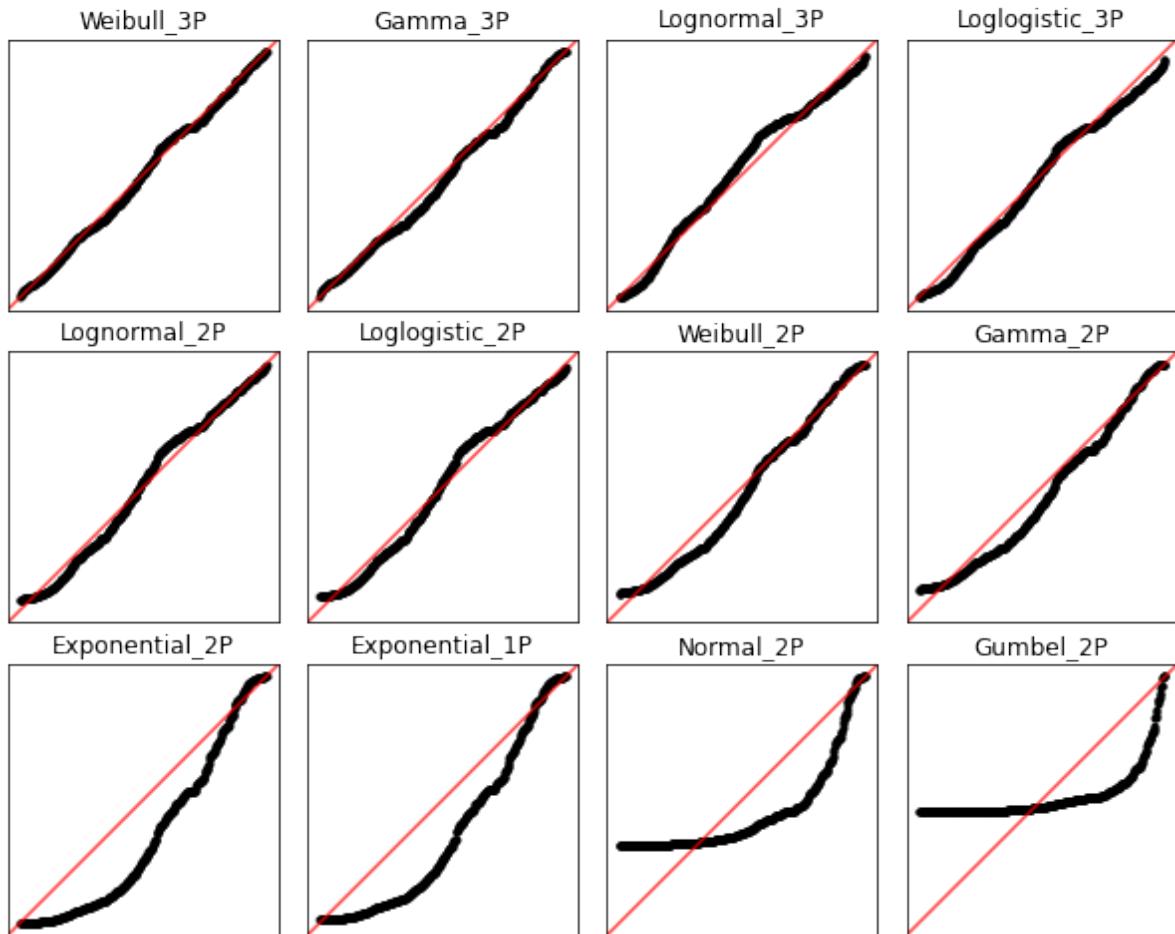


Figura 5.9: Probability-Probability plot semiparametrico

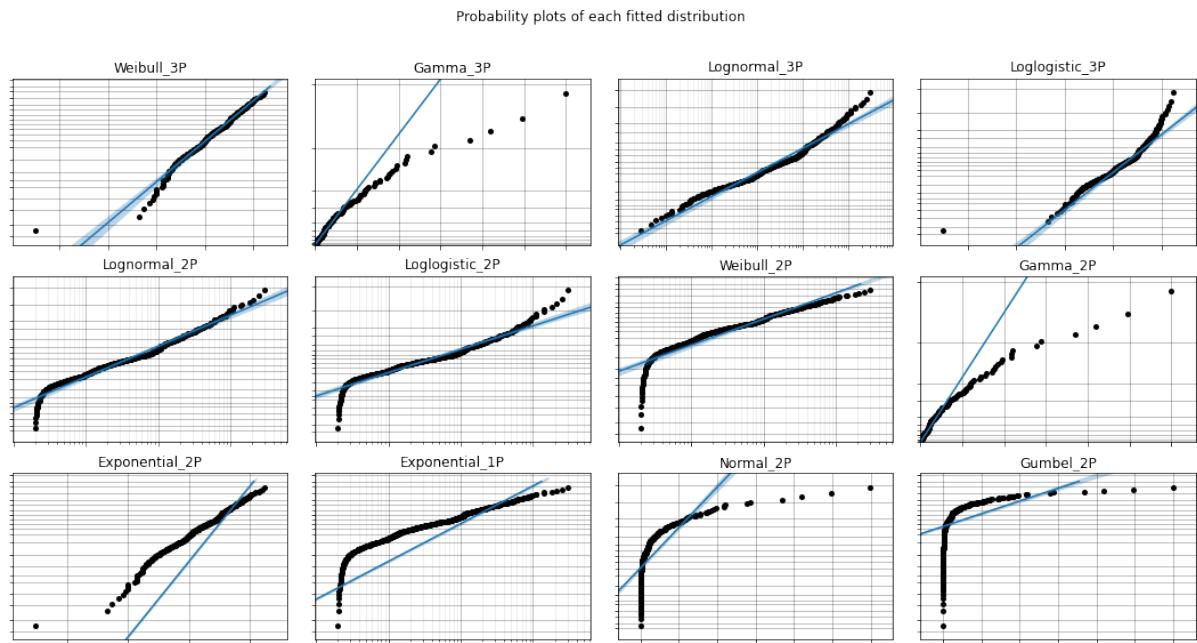


Figura 5.10: Probability plot per ogni distribuzione

Il risultato dell'analisi condotta mostra come la migliore distribuzione che approssima al meglio le funzioni empiriche del sistema è la **Weibull** con 3 parametri:

- $\alpha = 7914.9$
- $\beta = 0.538518$
- $\gamma = 201$

Dal risultato del test di fitting sulla distribuzione dei TTF si possono evincere anche delle informazioni riguardanti la natura del fallimento: in genere, infatti, la distribuzione di Weibull può manifestare la presenza di fenomeni degenerativi.

Come ulteriore prova di questo risultato, si è quindi proceduto ad effettuare il test di Kolmogorov - Smirnov il quale, confrontando la CDF empirica con quella ipotizzata, accetta con un livello di significatività di del 95% l'ipotesi che i dati provengano da una distribuzione di Weibull con i parametri sopra specificati.

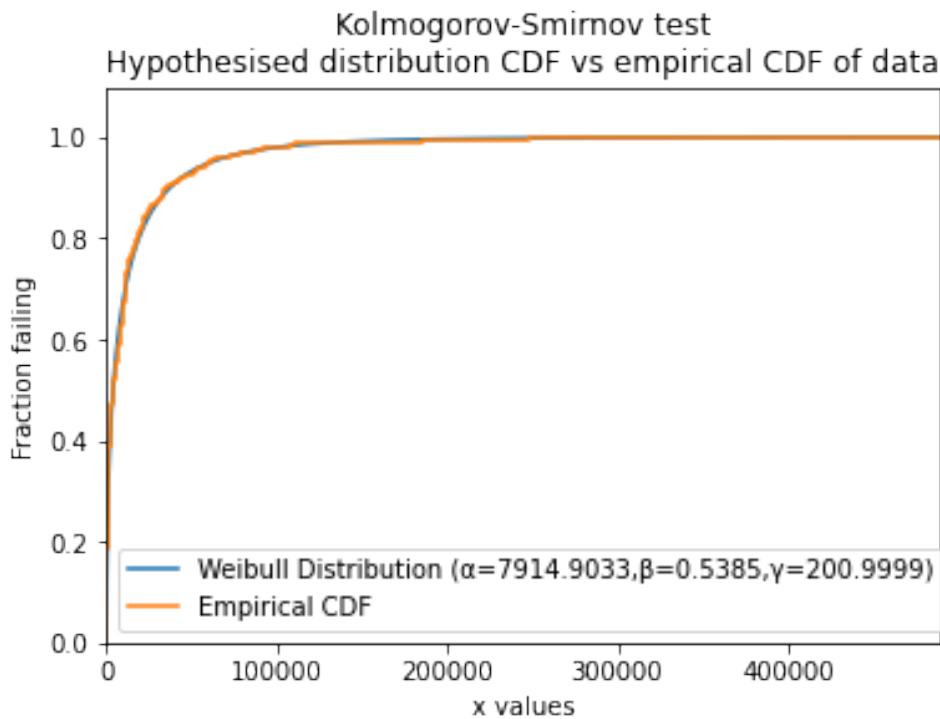


Figura 5.11: Test di Kolmogorov - Smirnov per distribuzione di Weibull

Nel dettaglio, l'approssimazione presenta un valore del coefficiente di determinazione R-squared pari a $R^2 = 0.9540972062314915$.

Per concludere lo step di Data Analysis, si ricava l'andamento della reliability empirica a partire dalla CDF empirica precedentemente calcolata:

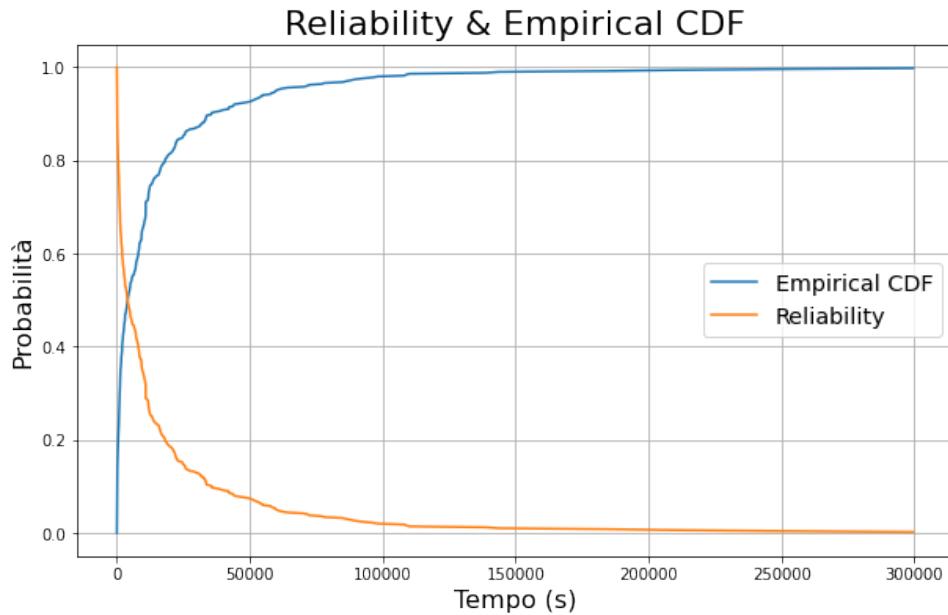


Figura 5.12: Reliability & CDF empiriche

5.4.4 Analisi per Sottosistema

Al fine di estrarre più informazioni dall'analisi condotta, si è scelto di rieseguire alcuni procedimenti precedentemente effettuati per ognuno dei sottosistemi dell'architettura.

Sottosistema Processore

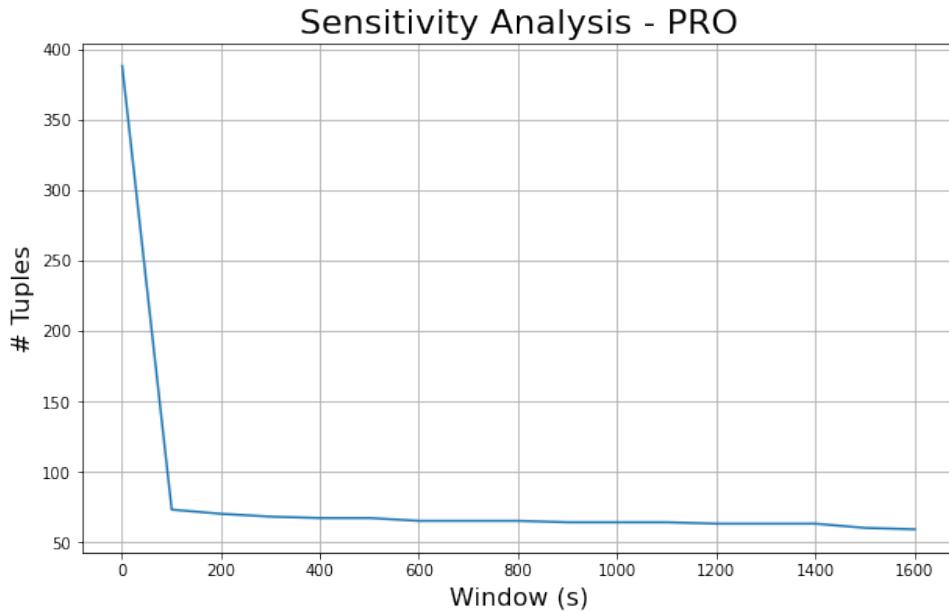


Figura 5.13: Sensitivity Analysis - PROCESSORE

Per il sottosistema Processore, la finestra di coalescenza scelta in precedenza per il sistema in generale è in realtà scorretta, in quanto il valore $W = 200$ rientra in una parte della funzione pressoché piatta, il che può essere sintomatico di un numero di collassi abbastanza cospicuo. Un valore più appropriato è costituito da $W = 100$.

Sottosistema Memoria

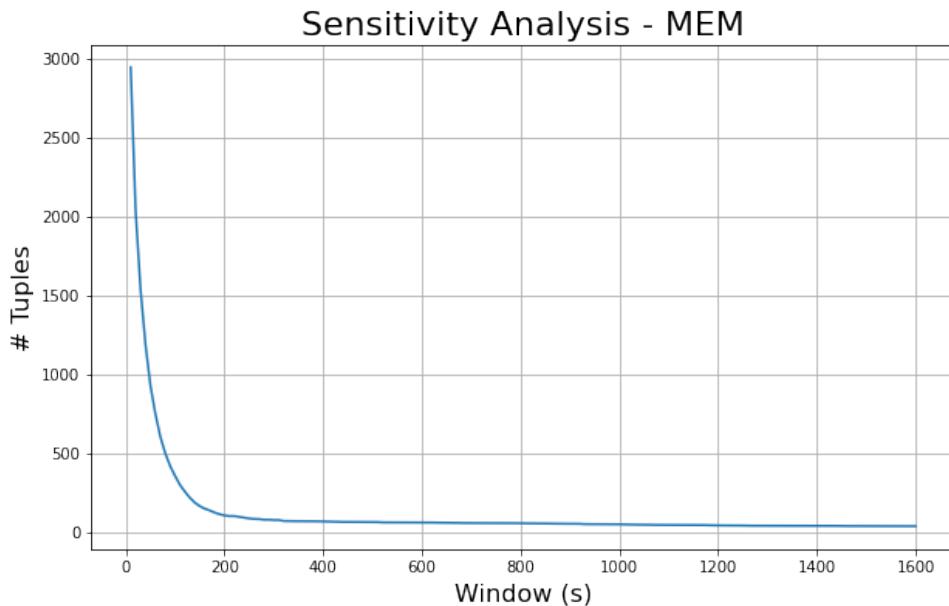


Figura 5.14: Sensitivity Analysis - MEMORIA

Per il sottosistema Memoria, la finestra di coalescenza scelta ($W = 200$) in precedenza per il sistema in generale può essere ritenuta corretta.

Sottosistema Device

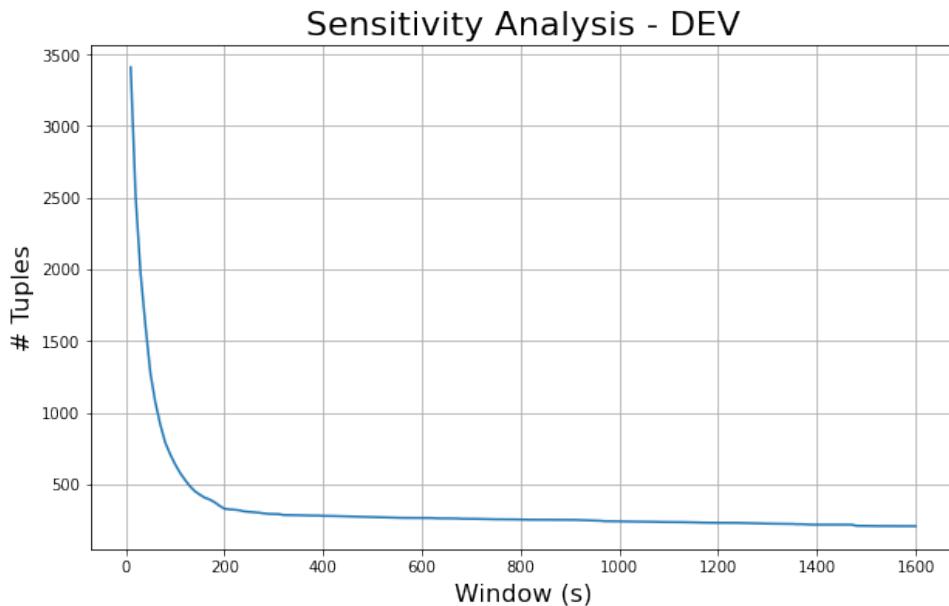


Figura 5.15: Sensitivity Analysis - DEVICE

Per il sottosistema Device, la finestra di coalescenza scelta ($W = 200$) in precedenza per il sistema in generale può essere ritenuta corretta.

Sottosistema Network

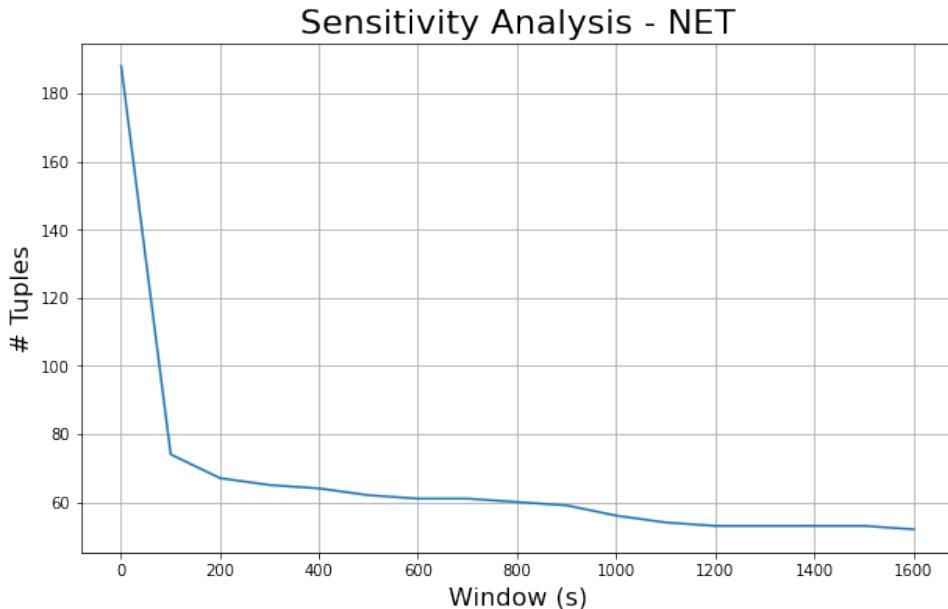


Figura 5.16: Sensitivity Analysis - NETWORK

Per il sottosistema Network, sebbene la funzione non sia facilmente interpretabile, un valore più appropriato di finestra di coalescenza potrebbe aggralarsi sul valore $W = 150$.

Sottosistema I/O

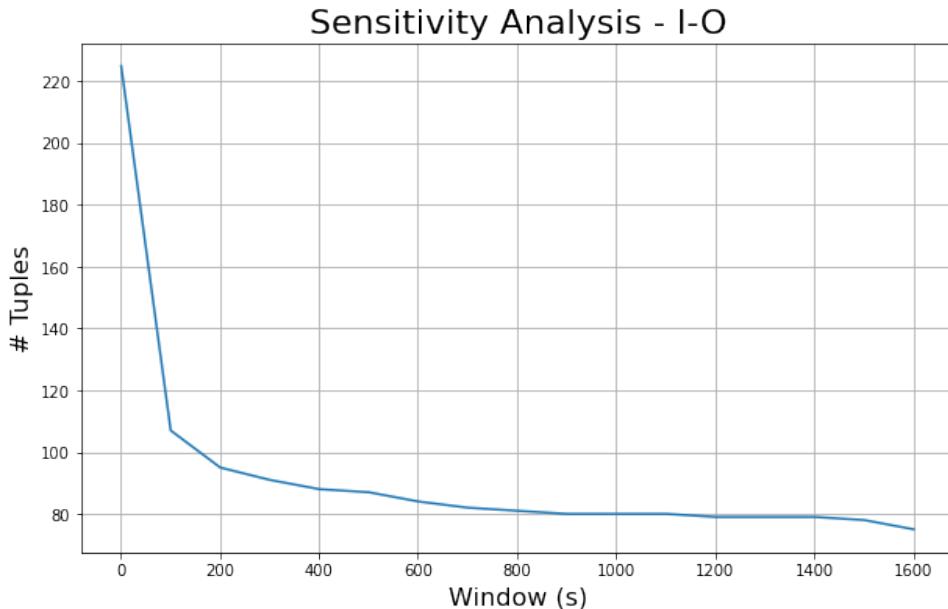


Figura 5.17: Sensitivity Analysis - I/O

Per il sottosistema Network, può essere nuovamente confermata la finestra di coalescenza iniziale.

L'analisi condotta ha dunque portato ai seguenti risultati:

Sottosistema	W	# tuple
DEV	200	95
MEM	200	104
I/O	200	95
NET	150	72
PRO	100	73

I risultati prodotti risultano essere coerenti con la distribuzione delle entry rispetto ai sottosistemi stessi, in quanto il numero di entry contenute nei sottosistemi NET e PRO (gli unici che richiedono una modifica della finestra) rappresentano una minima parte del numero di entry totale.

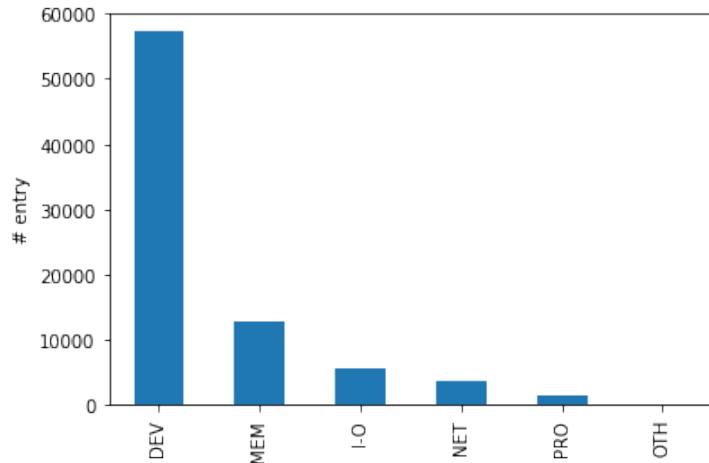


Figura 5.18: Numero di entry per ogni sottosistema

Con le finestre di coalescenza riportate, è stata calcolata la reliability empirica di seguito riportata, in cui si nota come il sottosistema meno reliable è DEV:

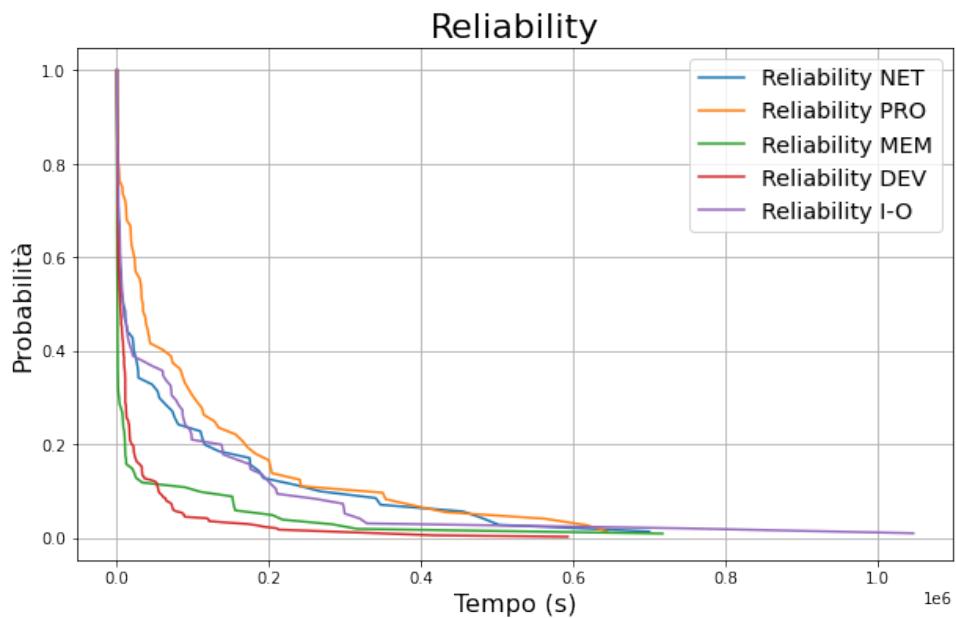


Figura 5.19: Reliability per ogni sottosistema

5.4.5 Analisi per Layer

L'analisi può essere ripetuta per i layer del sistema (*master*, *login*, *comput.* e *storage*).

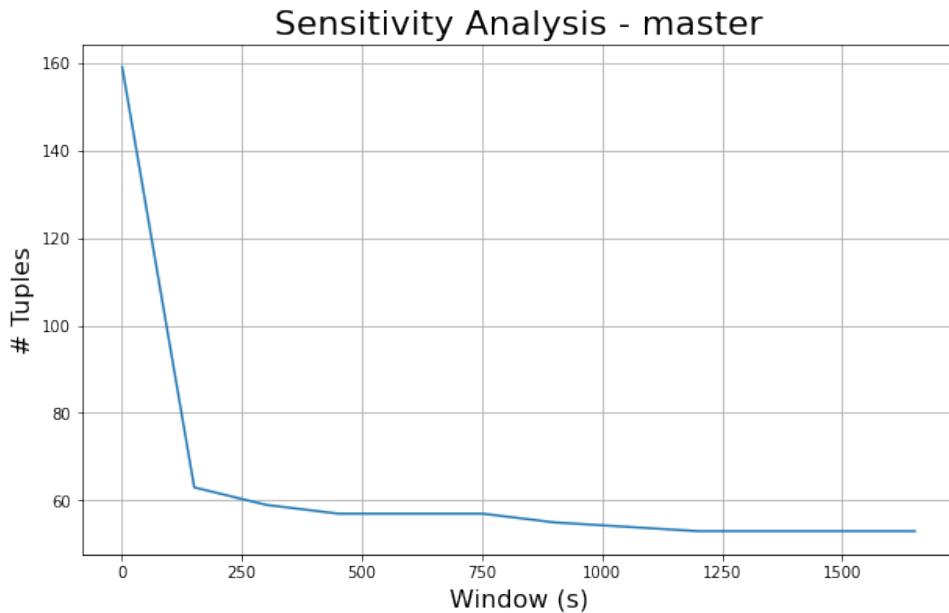


Figura 5.20: Sensitivity Analysis - Master Layer

La finestra di coalescenza più appropriata risulta essere $W = 150$.

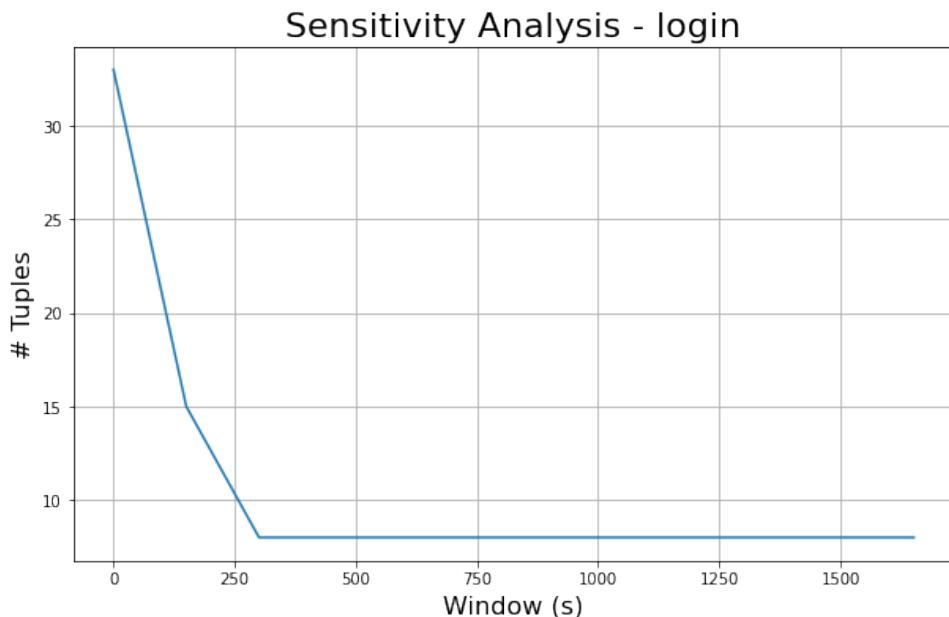


Figura 5.21: Sensitivity Analysis - Login Layer

La finestra di coalescenza più appropriata risulta essere $W = 150$.

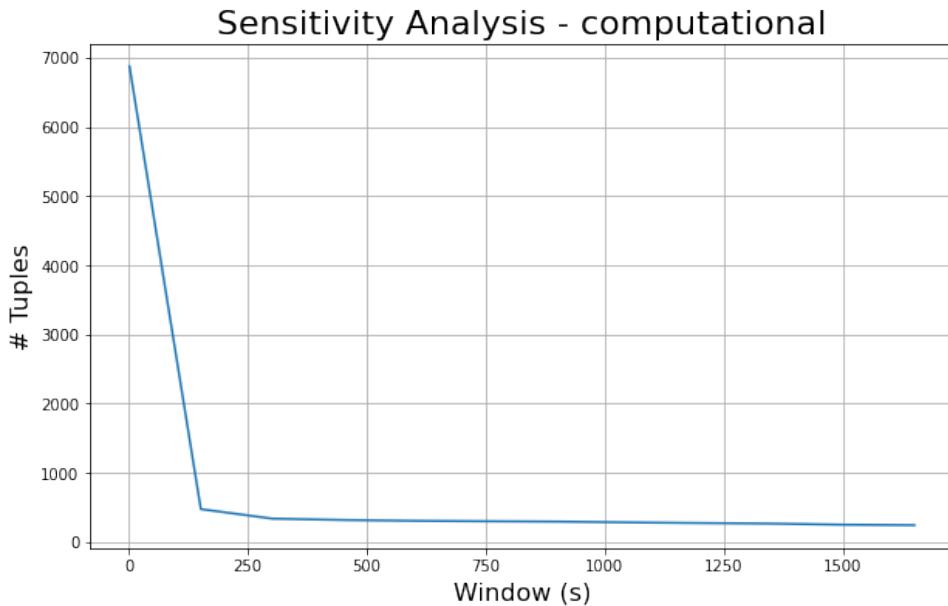


Figura 5.22: Sensitivity Analysis - Computational Layer

La finestra di coalescenza più appropriata risulta essere $W = 150$.

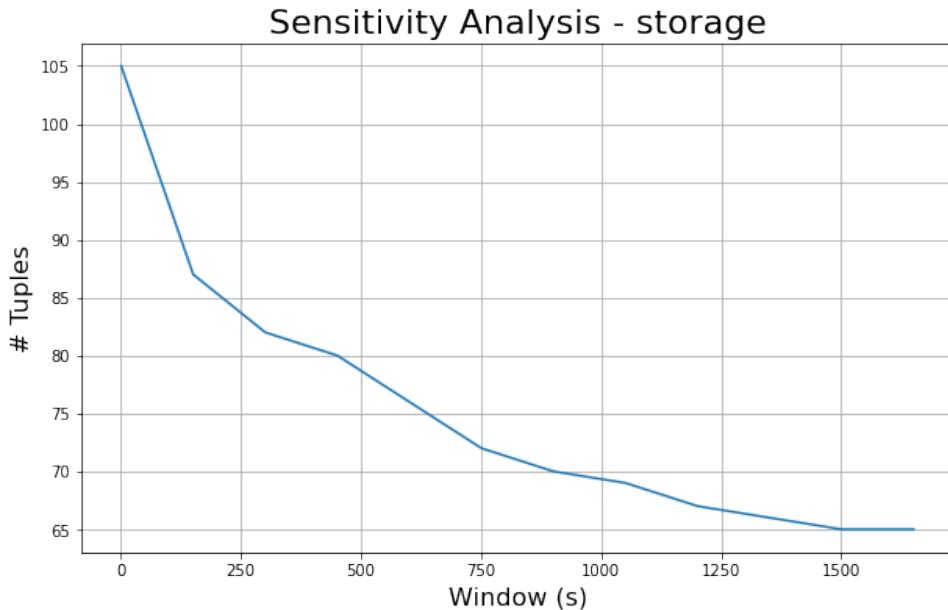


Figura 5.23: Sensitivity Analysis - Storage Layer

La finestra di coalescenza più appropriata è $W = 290$.

I risultati ottenuti sono riportati in tabella:

Layer	W	# tuple
Master	150	87
Login	150	15
Computational	150	477
Storage	290	82

Si riporta infine la reliability empirica per ogni layer, in cui si evidenzia come il layer meno reliable è quello computazionale:

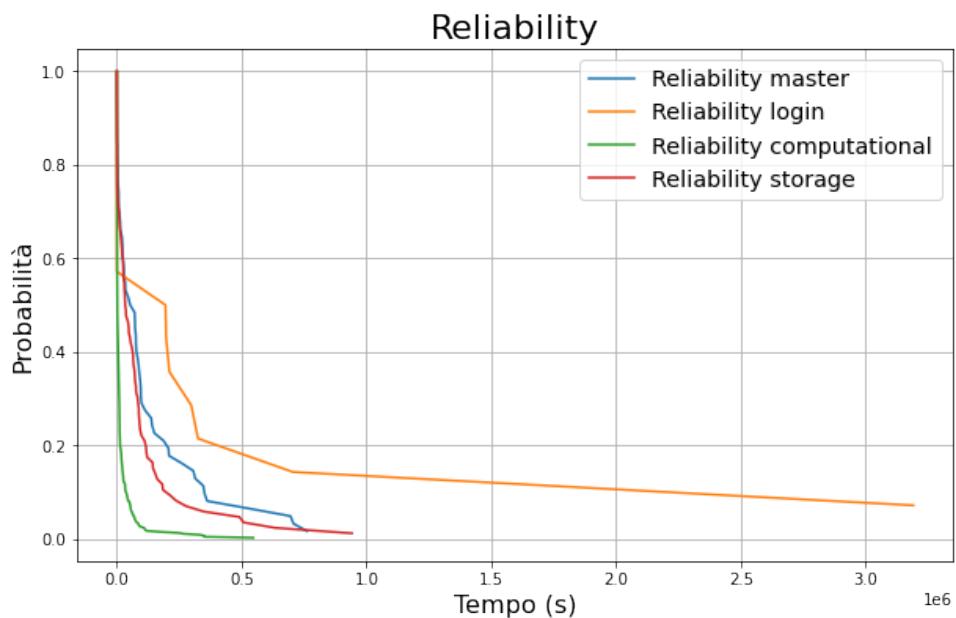


Figura 5.24: Reliability per ogni layer

5.4.6 Analisi dei nodi critici

Per effettuare un'analisi valida sui nodi del sistema, si identificano innanzitutto i nodi critici:

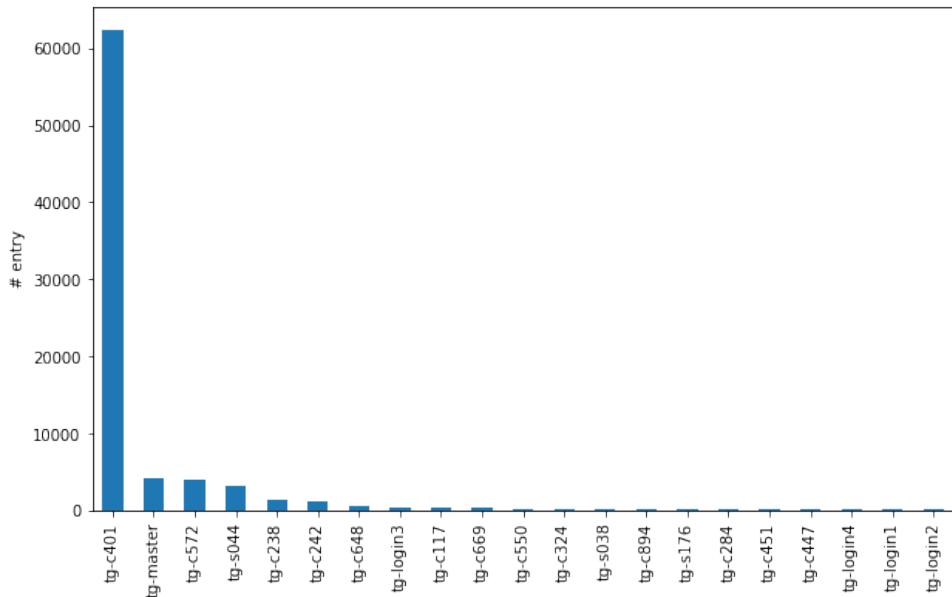


Figura 5.25: Nodi critici del sistema

Come risulta dal grafico appena mostrato, il nodo critico è senza dubbio *tg-c401*, come è emerso anche dall'analisi del bottleneck del sistema. Si è proceduto quindi a realizzare la sensitivity analysis per tale nodo:

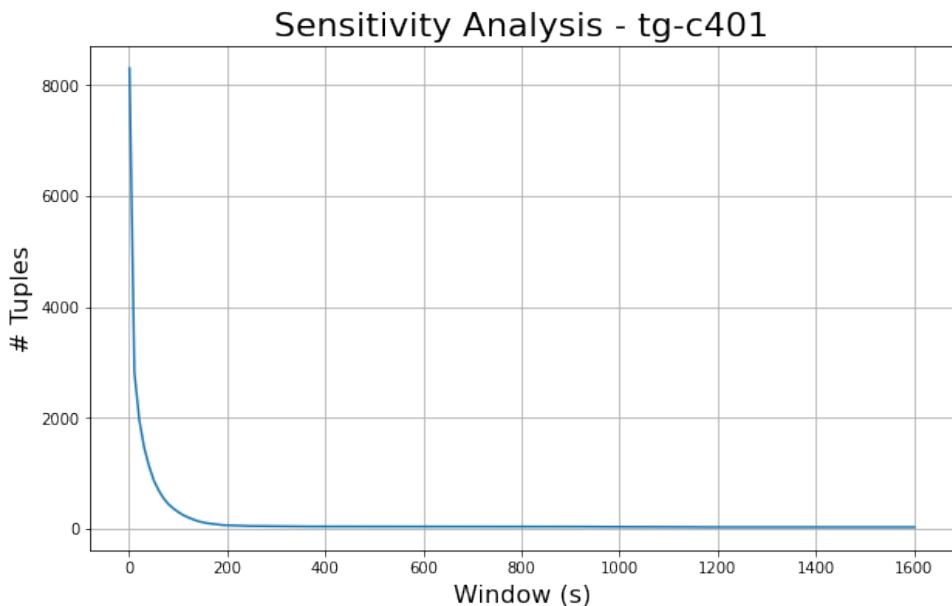


Figura 5.26: Sensitivity Analysis - *tg-c401*

Il valore di finestra di coalescenza più appropriato è probabilmente $W = 100$, a cui sono associate 59 tuple.

Di seguito si riporta la reliability del nodo stesso:

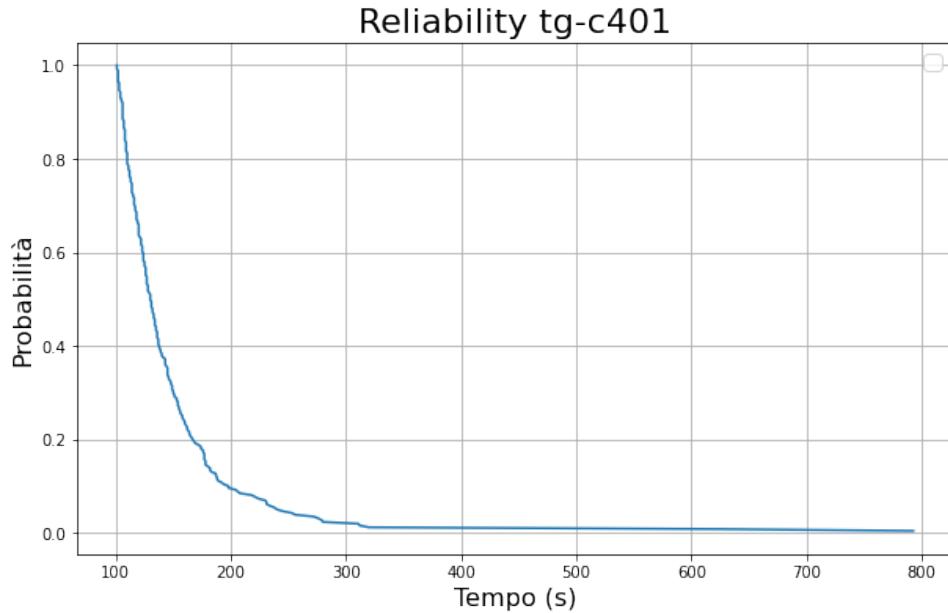


Figura 5.27: Reliability *tg-c401*

5.5 BlueGene/L

BlueGene è un supercalcolatore del *Lawrence Livermore National Labs*, di dimensioni maggiori del precedente sistema analizzato Mercury. L'organizzazione fisica del sistema prevede la presenza di *rack*, ognuno diviso in un certo numero di *midplane*, ognuno a sua volta costituito da *nodi*. Ciascun nodo possiede uno specifico numero di *compute card*, delle schede di calcolo costituite da 2 processori. Un insieme ristretto di nodi presenta 2 card aggiuntive dedicate all' *I/O*.

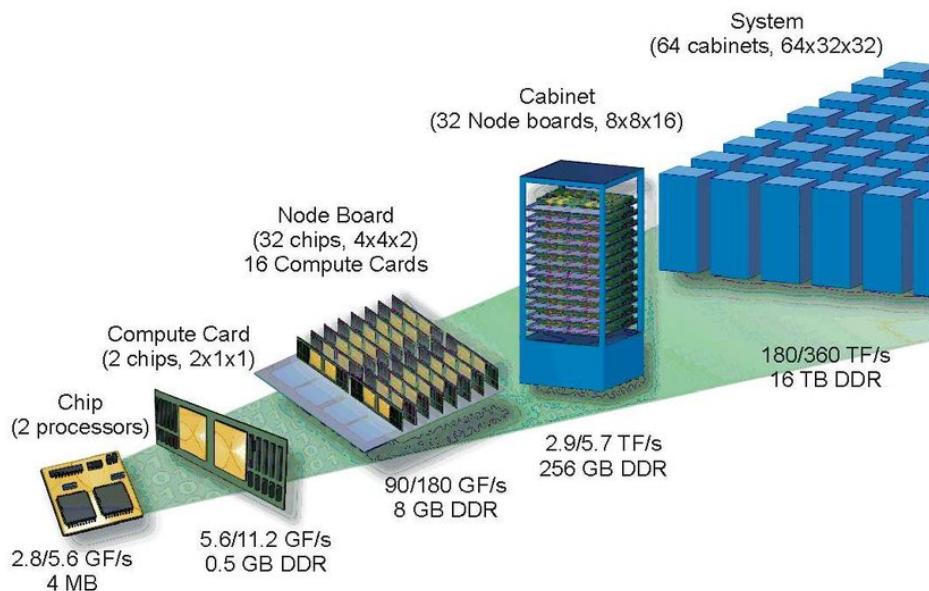


Figura 5.28: Architettura BlueGene/L

5.5.1 Log Analysis

Essendo dinanzi a file di log già collezionati e opportunamente filtrati, la FFDA in tal caso verrà condotta a partire dalla fase di data manipulation. Tuttavia, è necessario analizzare preventivamente i log filtrati, in modo da comprendere a pieno il problema in esame.

I log di BlueGene sono composti dalle seguenti informazioni:

- *Timestamp*
- *Originating node*: nodo che ha generato l'errore. Riporta le informazioni del componente fallito con una stringa del tipo *Rack-Midplane-Node*
- *Originating Card*: card all'interno del nodo causa dell'errore.
- *Message*: stringa riportante il messaggio di errore esteso.

	Timestamp	Node	Card	Message
0	1128621350	R00-M0-N0	J18-U01	Lustre mount FAILED : bglio2 : block_id : loc...
1	1128621350	R01-M1-N0	J18-U11	Lustre mount FAILED : bglio21 : block_id : lo...
2	1128621351	R07-M0-NC	J18-U01	Lustre mount FAILED : bglio124 : block_id : l...
3	1128621351	R00-M0-N4	J18-U01	Lustre mount FAILED : bglio4 : block_id : loc...
4	1128621351	R02-M0-N4	J18-U01	Lustre mount FAILED : bglio36 : block_id : lo...

Figura 5.29: BlueGene log

5.5.2 Data manipulation

Anche in questa casistica si è utilizzata la **coalescenza spaziale**, in modo tale da individuare una correlazione temporale tra le entry del log, le quali sono però appartenenti a componenti differenti del sistema.

L'algoritmo che permette di realizzare il **tupling** può essere schematizzato come segue:

```

if (ts( $x_{i+1}$ ) – ts( $x_i$ ) <  $W$ ) then
    aggiungi  $x_{i+1}$  alla tupla corrente
else
    genera una nuova tupla contenente  $x_{i+1}$ 

```

- una finestra temporale troppo grande potrebbe portare a dei collassi, raggruppando eventi relativi a fallimenti differenti nella stessa tupla
- una finestra temporale troppo piccola potrebbe causare dei troncamenti, scindendo eventi appartenenti allo stesso guasto in tuple differenti.

In prima battuta è stata realizzata nuovamente la **sensitivity analysis**, per individuare il miglior compromesso per la finestra di coalescenza W in modo da evitare un numero di collassi o troncamenti troppo spinto.

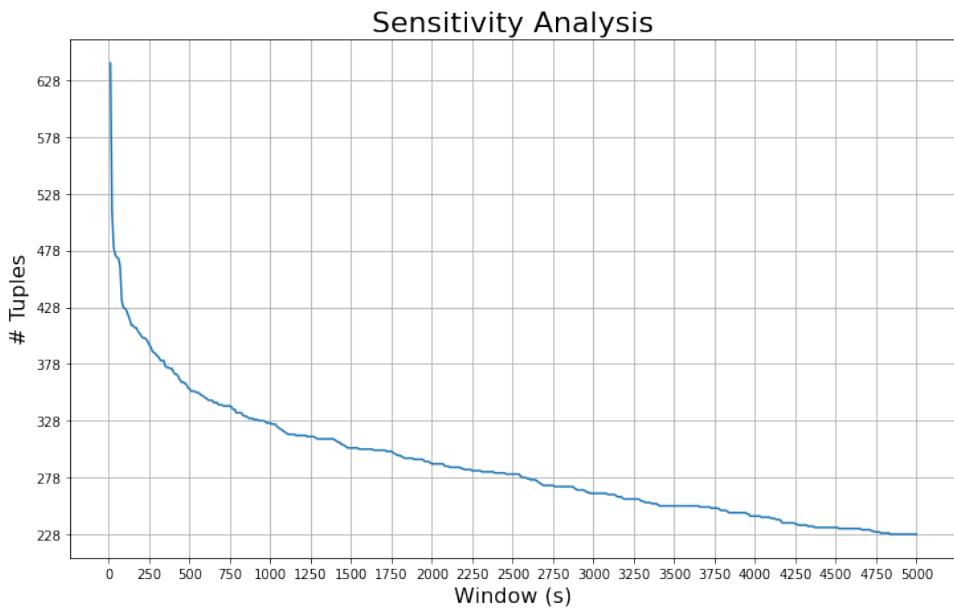


Figura 5.30: Sensitivity Analysis

Un buon compromesso per il valore della finestra temporale W è in genere quello appena successivo al gomito della curva. Per tale motivo si è scelto una finestra temporale $W = 500$.

Tramite tale valore della finestra di coalescenza, si è proceduto ad effettuare il tupling, il quale restituisce **356 tuple**. Per le varie tuple si è poi proceduto a calcolare alcune misurazioni statistiche:

- Numero di entry per tupla
- Starting point: timestamp della prima entry della tupla
- Ending point: timestamp dell'ultima entry della tupla
- Inter-arrival time: differenza temporale (in secondi) tra il timestamp della prima entry di una tupla e il timestamp dell'ultima entry della tupla precedente.
- Length: differenza tra il timestamp dell'ultima e della prima entry di ogni tupla
- Numero di nodi colpiti di fallimento, con relativo elenco di tali nodi

Tuple	Num_entries	Starting_point	Ending_point	Length	Num_nodes	Nodes
0	1	128	1128621350	1128621367	17	64 (R00-M0-N8, R05-M0-NC, R05-M1-NC, R07-M0-N8, R...
1	2	1024	1128641281	1128641411	130	512 (R44-M1-N4, R05-M0-NC, R50-M0-NC, R57-M1-N0, R...
2	3	23	1128702415	1128702464	49	1 (R61-M1-N2,)
3	4	9	1128719240	1128719316	76	1 (R06-M1-NE,)
4	5	14	1128747957	1128748024	67	1 (R23-M1-N0,)
...
351	352	32	1136133055	1136133059	4	16 (R35-M0-N4, R35-M1-NC, R35-M0-N8, R35-M1-N8, R...
352	353	264	1136287486	1136287526	40	65 (R21-M1-N0, R20-M1-NC, R24-M0-N4, R24-M1-N4, R...
353	354	2	1136318181	1136318198	17	2 (R52-M0-N8, R76-M0-N8)
354	355	333	1136322566	1136322676	110	270 (R10-M1-NC, R54-M0-N4, R50-M0-NC, R42-M0-NC, R...
355	356	8	1136390405	1136390405	0	8 (R00-M0-NC, R20-M1-NC, R36-M0-NC, R31-M0-NC, R...

	Tuples	Interarrivals
0	Tuple 1-2	19914
1	Tuple 2-3	61004
2	Tuple 3-4	16776
3	Tuple 4-5	28641
4	Tuple 5-6	46370
...
350	Tuple 351-352	175640
351	Tuple 352-353	154427
352	Tuple 353-354	30655
353	Tuple 354-355	4368
354	Tuple 355-356	67729

Figura 5.31: Statistiche delle tuple

Analisi del bottleneck

Tramite le statistiche appena illustrate si sono ricercati dei possibili fallimenti particolarmente critici nel sistema, confrontando tra loro il numero entry per ogni tupla.

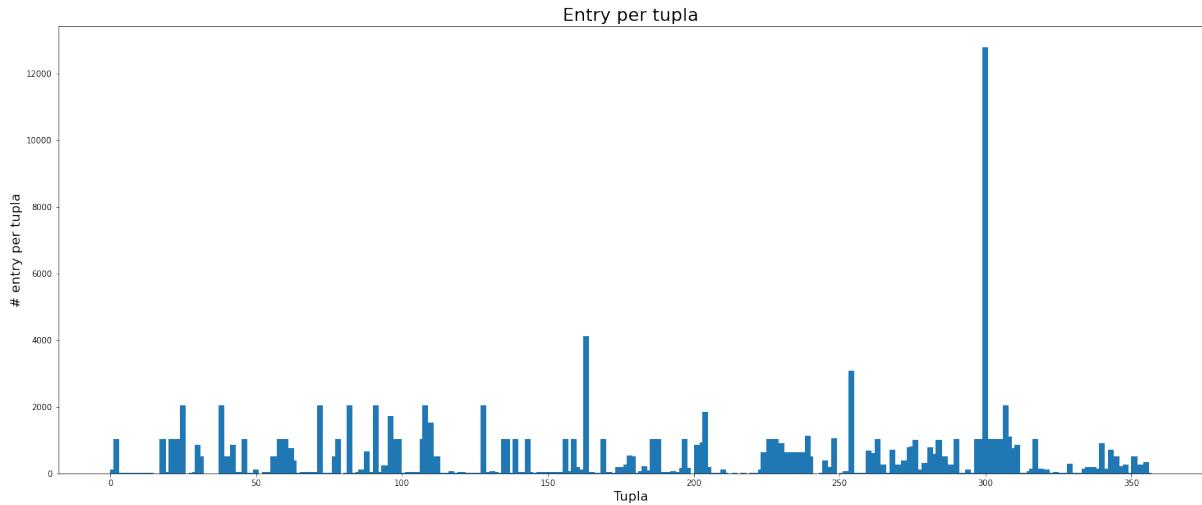


Figura 5.32: Numero di entry per ogni tupla

Da tale grafico ne consegue come i fallimenti del sistema siano in realtà equamente distribuiti nelle varie tuple, eccezion fatta per la tupla 300, la quale sfiora le 12 mila entry, circa il 10% di tutti gli eventi totali del log.

Per approfondire, si è realizzato il seguente grafico che evidenzia la somma cumulativa del numero di entry di ogni tupla:

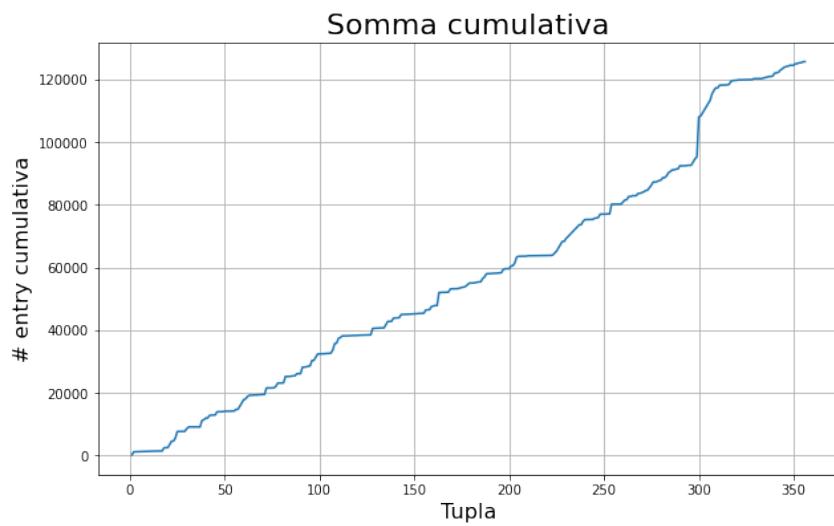


Figura 5.33: Somma cumulativa della distribuzione di entry

L'andamento della somma cumulativa conferma l'intuizione precedente, in quanto il grafico presenta avere un andamento pressoché lineare. L'unica zona di ampia pendenza

è proprio in corrispondenza della tupla 300. Tale tupla potrebbe quindi contenere delle informazioni sul bottleneck del sistema. Trasformando l'intervallo dei timestamp della tupla 300 in formato data, si ottiene il seguente intervallo temporale:

- Data inizio: 2005-12-09 21:07:05
- Data fine: 2005-12-09 21:55:19

L'intervallo temporale in cui si sono verificati molti fallimenti è di appena **48 minuti, 14 secondi**. Tale durata è di fatto quasi nulla se paragonata alla totalità della durata temporale coperta dai log, di 89 giorni.

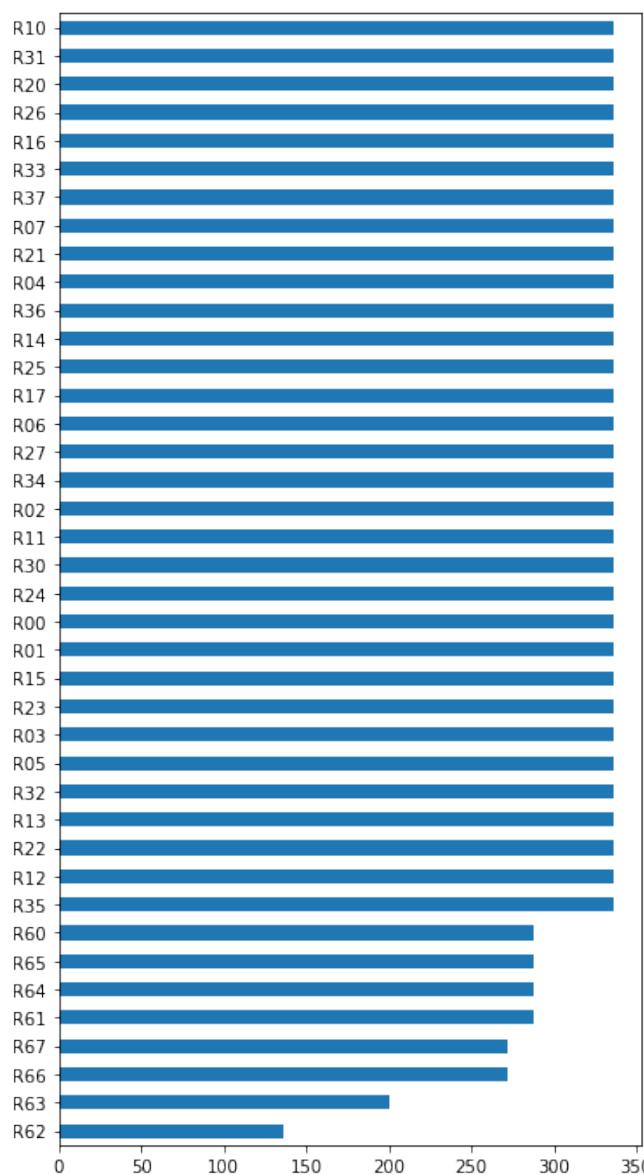


Figura 5.34: Eventi di errore per rack

Dal grafico della distribuzione dei fallimenti nel bottleneck tra i vari rack, si può notare come in realtà i fallimenti siano piuttosto diffusi in tutto il sistema e non isolati ad una parte di esso.

		Message Counts
0	ciod: Error loading /bgl/apps/SWL/stability/NEWS05/news05_DD: invalid or missing program image	1384
1	ciod: Error loading /bgl/apps/SWL/stability/MDCASK/WORK/1415/inferno: invalid or missing program image	512
2	ciod: Error loading /p/gb1/stella/SPPM/1268/sppm_DD: invalid or missing program image	512
3	ciod: Error loading /bgl/apps/SWL/stability/MDCASK/WORK/1295/inferno: invalid or missing program image	512
4	ciod: Error loading /p/gb1/stella/SPASM/1308/spasm_DD: invalid or missing program image	512
5	ciod: Error loading /p/gb1/stella/UMT2K/1372/umt2k_DD: invalid or missing program image	512
6	ciod: Error loading /bgl/apps/SWL/stability/MDCASK/WORK/1378/inferno: invalid or missing program image	512
7	ciod: Error loading /bgl/apps/SWL/stability/HPL/WORK/1405./hpl_DD: invalid or missing program image	512
8	ciod: Error loading /p/gb1/stella/SPPM/1366/sppm_DD: invalid or missing program image	512
9	ciod: Error loading /bgl/apps/SWL/stability/HPL/WORK/1398./hpl_DD: invalid or missing program image	512
10	ciod: Error loading /bgl/apps/SWL/stability/MDCASK/WORK/1288/inferno: invalid or missing program image	512
11	ciod: Error loading /bgl/apps/SWL/stability/MDCASK/WORK/1339/inferno: invalid or missing program image	512
12	ciod: Error loading /p/gb1/stella/SPASM/1352/spasm_DD: invalid or missing program image	512
13	ciod: Error loading /p/gb1/stella/UMT2K/1328/umt2k_DD: invalid or missing program image	512
14	ciod: Error loading /p/gb1/stella/SPPM/1356/sppm_DD: invalid or missing program image	512
...
54	ciod: Error loading /bgl/apps/SWL/stability/HPL/WORK/1386/hpl_DD: invalid or missing program image	16
55	ciod: Error loading /p/gb1/stella/SPPM/1361/sppm_DD: invalid or missing program image	16
56	ciod: Error loading /p/gb1/stella/SPASM/1347/spasm_DD: invalid or missing program image	16
57	ciod: Error loading /bgl/apps/SWL/stability/MDCASK/WORK/1410/inferno: invalid or missing program image	16
58	ciod: Error loading /bgl/apps/SWL/stability/UMT2K/WORK/1367/umt2k_DD: invalid or missing program image	16
59	ciod: Error loading /bgl/apps/SWL/stability/HPL/WORK/1400./hpl_DD: invalid or missing program image	16
60	ciod: Error loading /bgl/apps/SWL/stability/MDCASK/WORK/1409/inferno: invalid or missing program image	8
61	ciod: Error loading /bgl/apps/SWL/stability/HPL/WORK/1399./hpl_DD: invalid or missing program image	8
62	ciod: Error loading /bgl/apps/SWL/stability/HPL/WORK/1495./hpl_DD: invalid or missing program image	8
63	ciod: Error loading /p/gb1/stella/SPPM/1360/sppm_DD: invalid or missing program image	8
64	ciod: Error loading /p/gb1/stella/RAPTOR/1419/raptor: invalid or missing program image	8
65	ciod: Error loading /bgl/apps/SWL/stability/HPL/WORK/1392./hpl_DD: invalid or missing program image	8
66	ciod: Error loading /bgl/apps/SWL/stability/MDCASK/WORK/1379/inferno: invalid or missing program image	8
67	ciod: Error loading /p/gb1/stella/RAPTOR/1420/raptor: invalid or missing program image	8
68	ciod: Error loading /bgl/apps/SWL/stability/MDCASK/WORK/1408/inferno: invalid or missing program image	8

Figura 5.35: Messaggi di errore della tupla 300

Dai messaggi di errore contenuti nella tupla 300, si evince come la totalità di essi sono tutti registrati dal *node daemon CIOD* (Console I-O Daemon), il quale è in esecuzione su ogni nodo di I-O e rappresenta una sorta di ponte tra i nodi computazionali e il mondo esterno. Probabilmente, il bottleneck del sistema è rappresentato proprio dal sottosistema di I-O.

La prova di tale intuizione è ottenibile dalla distribuzione di eventi in relazioni alle Card, operazione che evidenzia come le card con più errori sono **J18-U01** e **J18-U11**, entrambe dedicate all'I-O del sistema.

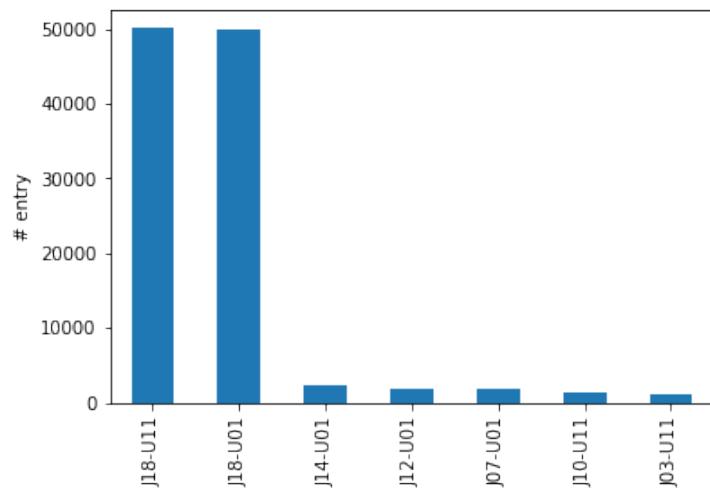


Figura 5.36: Entry per Card

Analisi dei collassi

L'analisi dei collassi prevede di stimare il numero di collassi commessi nell'operazione di tupling: difatti, se la finestra temporale scelta è troppo ampia, eventi relativi a guasti diversi possono essere accorpati nella stessa tupla. Si è proceduto ad effettuare l'analisi dei collassi andando a rilevare il numero di tuple in cui sono contenuti eventi relativi a più nodi di origine. Sono ben 238 le tuple che presentano questa caratteristica, le quali rappresentano oltre il 65% delle tuple totali. Anche l'analisi dei collassi eseguita rispetto ai rack ha portato ad una percentuale di collassi piuttosto alta, circa il 50%.

Tuttavia, l'analisi tale analisi è fortemente influenzata dal fatto che la maggior parte delle tuple con collassi che presentano precisamente 2 nodi di origine degli errori, presentano l'accoppiata di nodi di origine *J18-U01* e *J18-U11*. Tali nodi, tuttavia, rappresentano i nodi di I-O. Per quanto detto in precedenza, dunque, si ha che tali nodi sono sostanzialmente legati dallo stesso fallimento, ossia quello del sottosistema di I-O. Accorpando questa tipologia di fallimenti, la percentuale di collassi si riduce dal 65% al 15%.

Analisi dei troncamenti

L'analisi dei troncamenti consiste invece nell'individuare gli eventi dovuti allo stesso fallimento che sono stati divisi in tuple diverse, a causa di una finestra di coalescenza troppo piccola. Tale analisi può essere condotta andando a ricercare eventi dovuti allo stesso nodo in tuple consecutive. Un'analisi di questo tipo porta ad una stima della truncation pari circa al 40%. Tale stima è piuttosto alta: una possibile causa può essere dovuta al fatto che, per come è stato costruito l'algoritmo di tupling, la distanza tra due tuple consecutive potrebbe essere piuttosto ampia. Si è quindi preventivamente individuato il valore corrispondente al *0.20-quantile* dei tempi di interarrivo, pari a 2236 secondi, in modo da andare a considerare l'insieme di tuple consecutive il cui tempo di inter-arrivo sia inferiore al valore di quantile indicato. Solo successivamente a questa operazione si è quindi proceduto con l'effettiva verifica di tuple consecutive affette da errori relativi allo stesso nodo. Ne consegue che circa l'11.5% delle tuple totali potrebbe essere colpito da truncation.

5.5.3 Data Analysis

L'ultimo passo del processo di FFDA consiste nell'analisi statistica dei dati per poterne estrarre informazioni riguardanti la reliability empirica del sistema. Proprio per ricavare la stima della reliability, si è iniziato col determinare la distribuzione di probabilità(*pdf*) dei tempi di interarrivo delle tuple, rappresentanti di fatto il TTF; a partire dalla pdf empirica si è potuto poi calcolare la funzione di distribuzione cumulativa(*CDF*), che a sua volta rappresenta la probabilità che il sistema fallisca in un certo istante t ; finalmente, si può ricavare la reliability empirica del sistema come: $R(t) = 1 - CDF(t)$

Tale procedimento è stato realizzato tramite la libreria python *reliability*, la quale è in grado di realizzare in automatico anche il successivo step di **Goodness Of Fitting (*Gof*)**, al fine di verificare quale delle distribuzioni note è in grado di approssimare al meglio le funzioni di probabilità ottenute. Le distribuzioni note utilizzate per il fitting sono: Weibull, log-normale, Gamma, Esponenziale, log-logistica, Gumbel (tutte con diverse combinazioni di parametri).

Di seguito è riportato il risultato del procedimento:

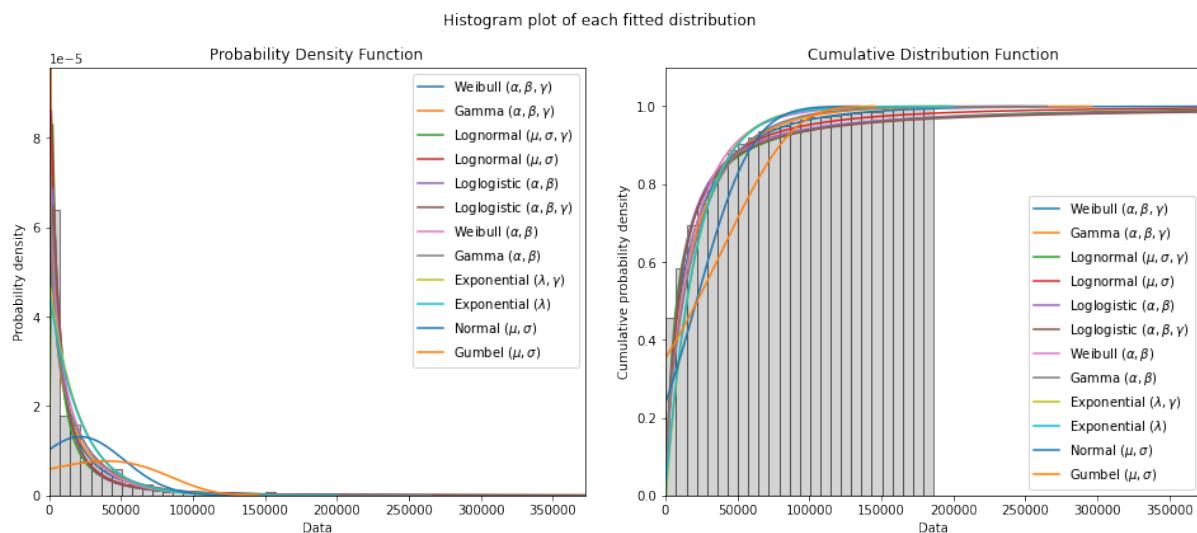


Figura 5.37: *pdf* & *CDF* empiriche

Semi-parametric Probability-Probability plots of each fitted distribution
Parametric (x-axis) vs Non-Parametric (y-axis)

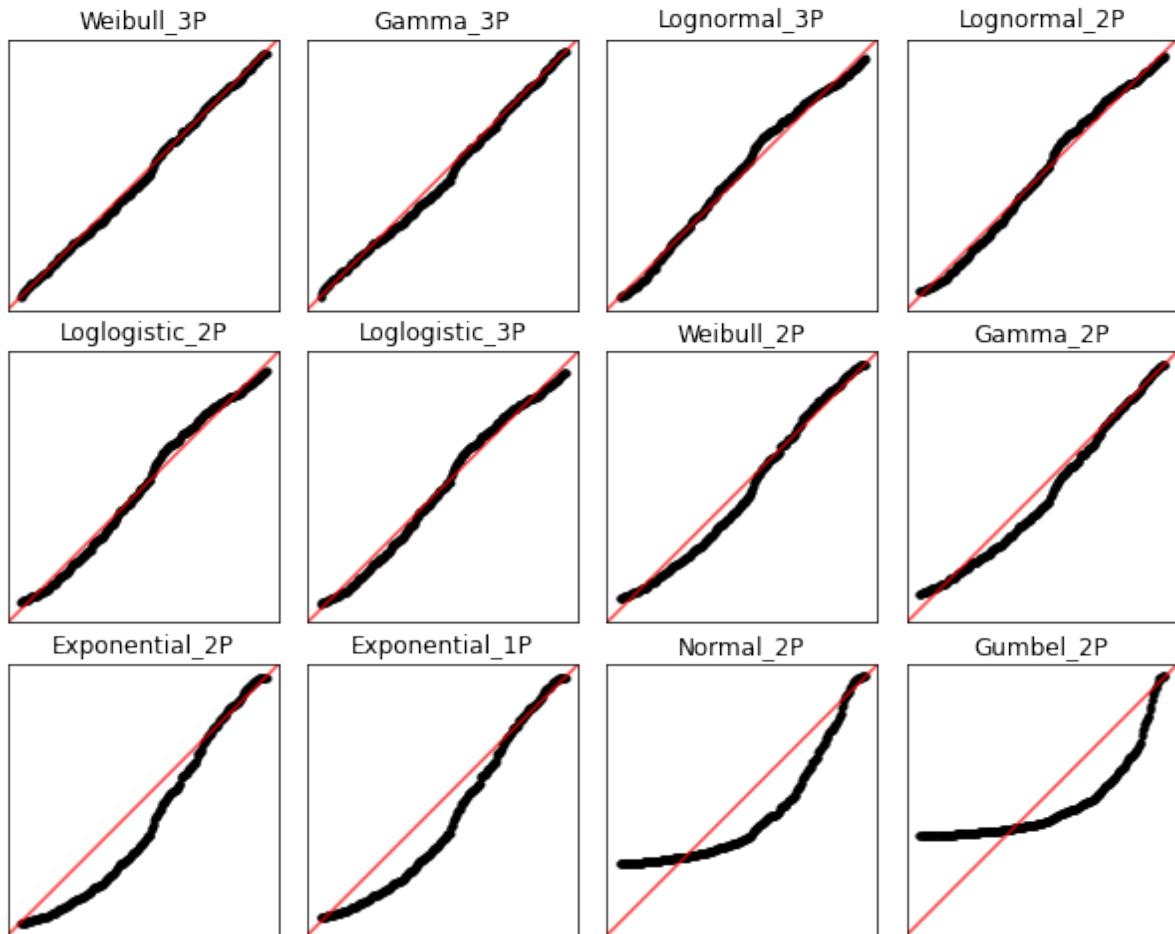


Figura 5.38: Probability-Probability plot semiparametrico

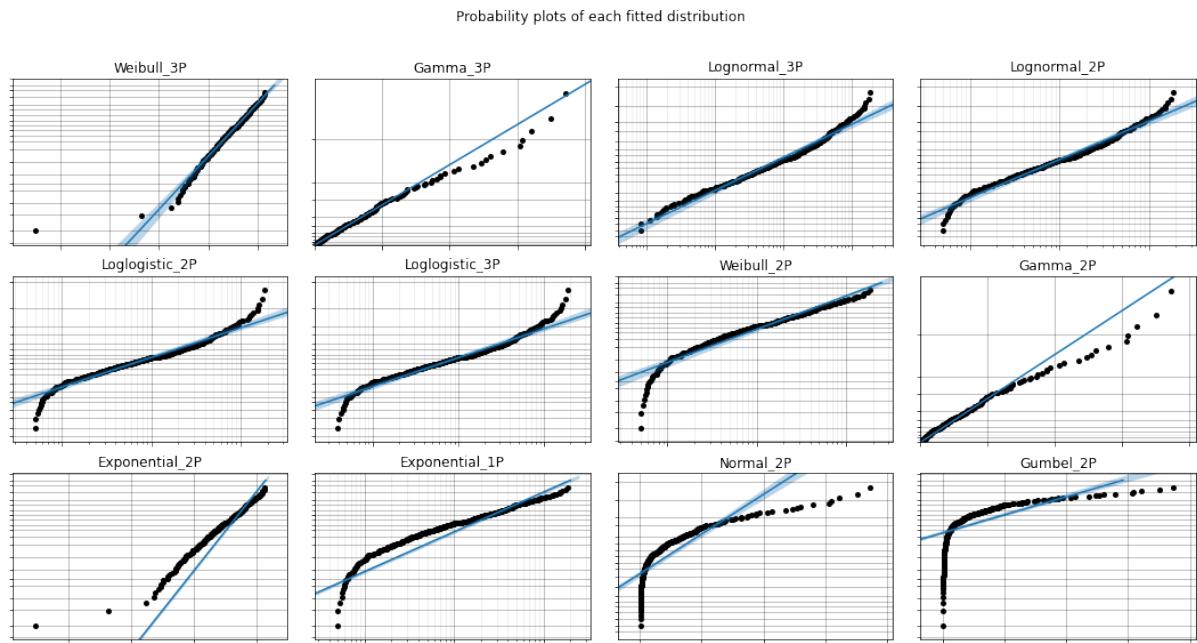


Figura 5.39: Probability plot per ogni distribuzione

Il risultato dell'analisi condotta mostra come la migliore distribuzione che approssima al meglio le funzioni empiriche del sistema è la **Weibull** con 3 parametri:

- $\alpha = 16307.9$
- $\beta = 0.67234$
- $\gamma = 503$

Dal risultato del test di fitting sulla distribuzione dei TTF si possono evincere anche delle informazioni riguardanti la natura del fallimento: in genere, infatti, la distribuzione di Weibull può manifestare la presenza di fenomeni degenerativi.

Come ulteriore prova di questo risultato, si è quindi proceduto ad effettuare il test di Kolmogorov - Smirnov il quale, confrontando la CDF empirica con quella ipotizzata, accetta con un livello di significatività di del 95% l'ipotesi che i dati provengano da una distribuzione di Weibull con i parametri sopra specificati.

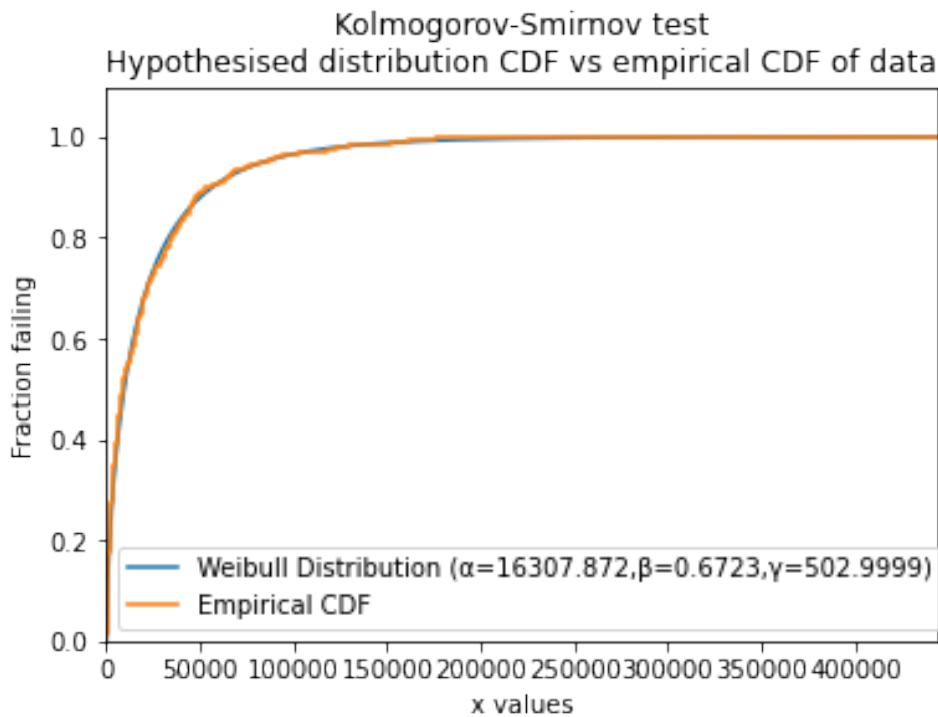


Figura 5.40: Test di Kolmogorov - Smirnov per distribuzione di Weibull

In particolare, il valore di R-squared risultante dall'approssimazione è pari a $R^2 = 0.9837567491816327$.

Per concludere lo step di Data Analysis, si ricava l'andamento della reliability empirica a partire dalla CDF empirica precedentemente calcolata:

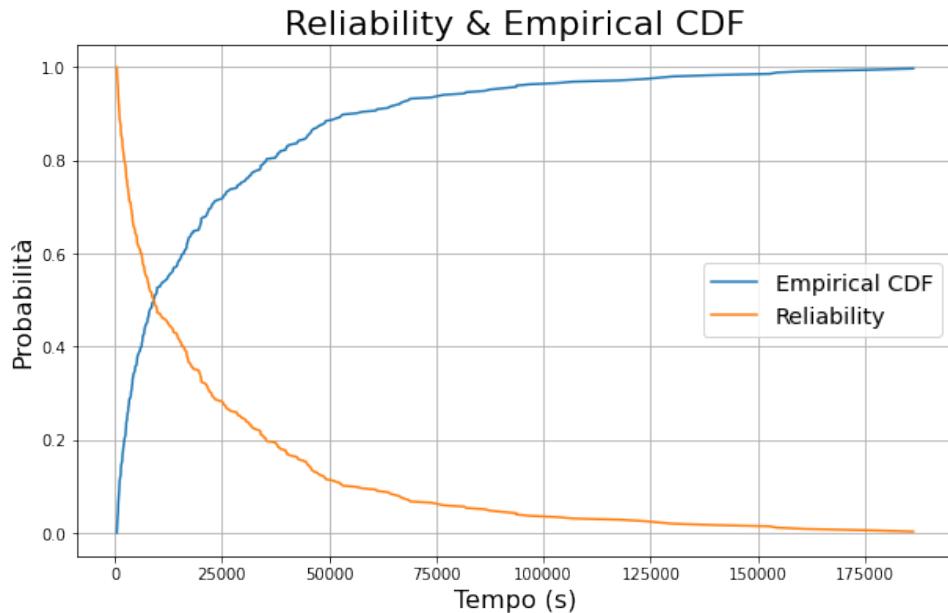


Figura 5.41: Reliability & CDF empiriche

5.5.4 Analisi per Rack

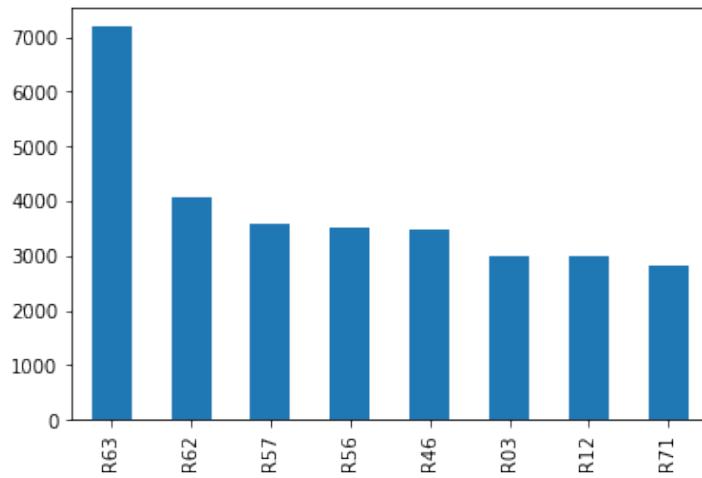


Figura 5.42: Rack più critici

Al fine di estrarre più informazioni dall'analisi condotta, si è scelto di rieseguire alcuni procedimenti precedentemente effettuati per i 5 rack più critici dell'architettura.

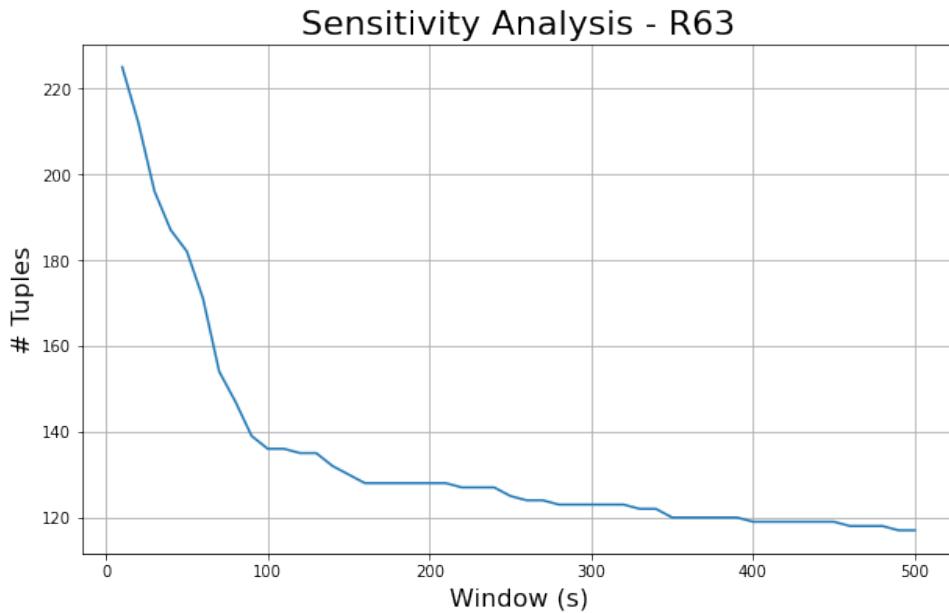


Figura 5.43: Sensitivity Analysis - R63

La finestra di coalescenza più appropriata è $W = 100$.

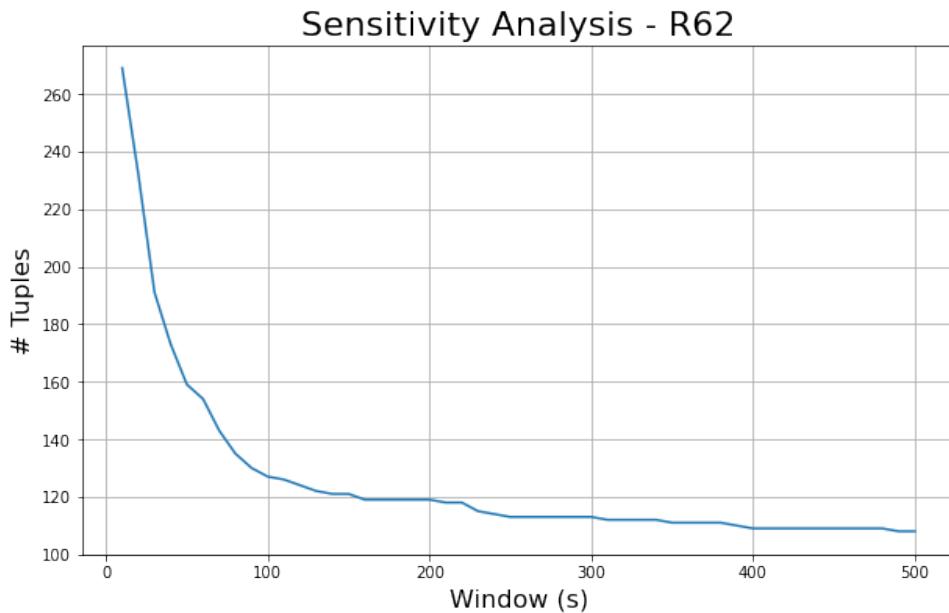


Figura 5.44: Sensitivity Analysis - R62

La finestra di coalescenza più appropriata è $W = 100$.

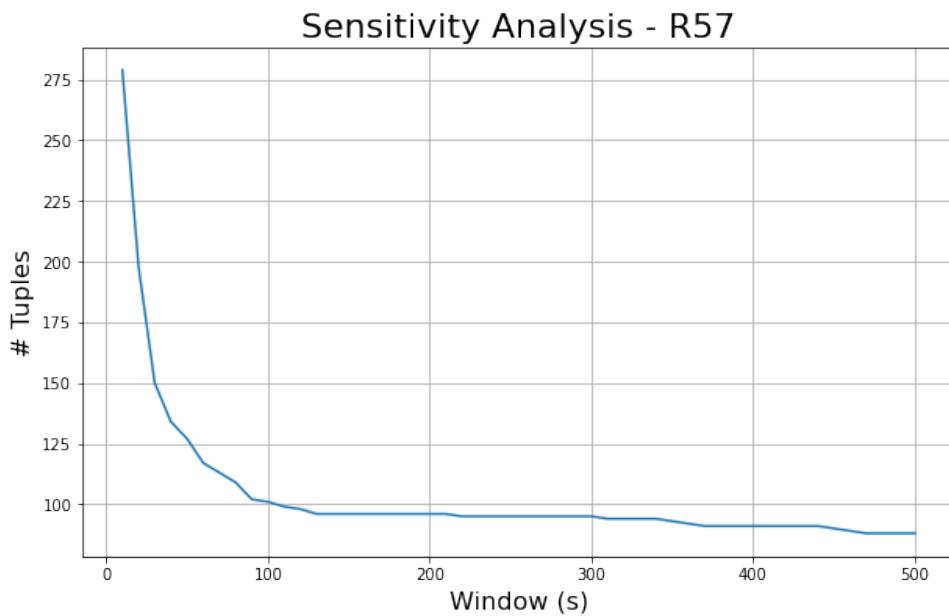


Figura 5.45: Sensitivity Analysis - R57

La finestra di coalescenza più appropriata è $W = 100$.

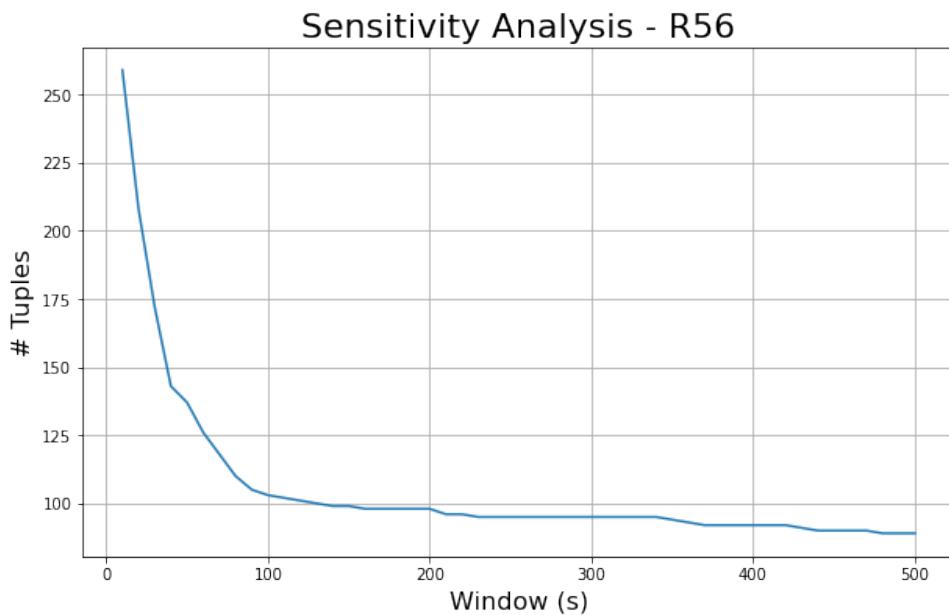


Figura 5.46: Sensitivity Analysis - R56

La finestra di coalescenza più appropriata è $W = 100$.

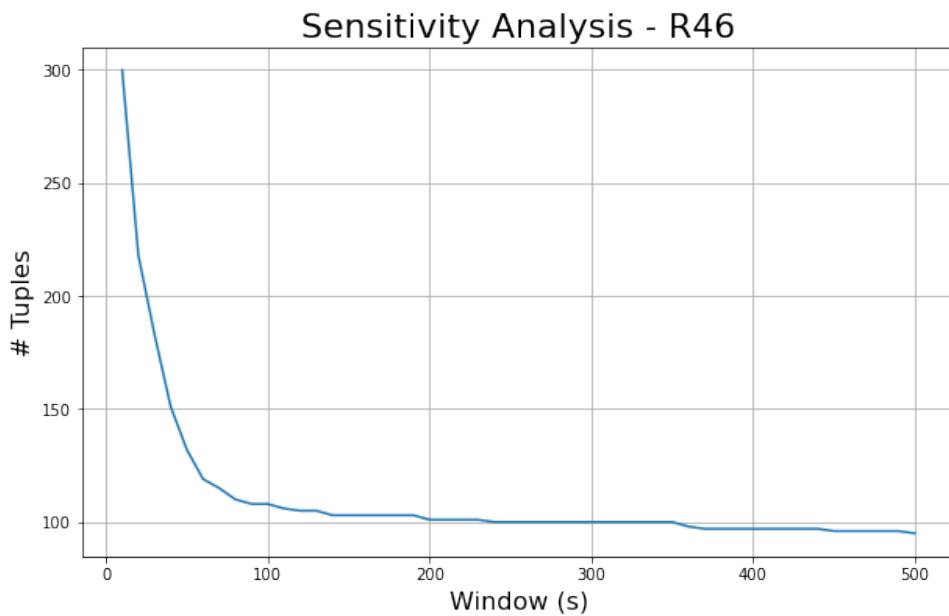


Figura 5.47: Sensitivity Analysis - R46

La finestra di coalescenza più appropriata è $W = 75$.

L'analisi condotta ha dunque portato ai seguenti risultati:

Rack	W	# tuple
R63	100	136
R62	100	127
R57	100	101
R56	100	103
R46	75	113

Con le finestre di coalescenza riportate, è stata calcolata la reliability empirica di seguito riportata. Le varie reliability sono pressoché simili tra i vari rack, tuttavia il rack meno reliable è R63, il quale è anche il rack che registra più eventi:

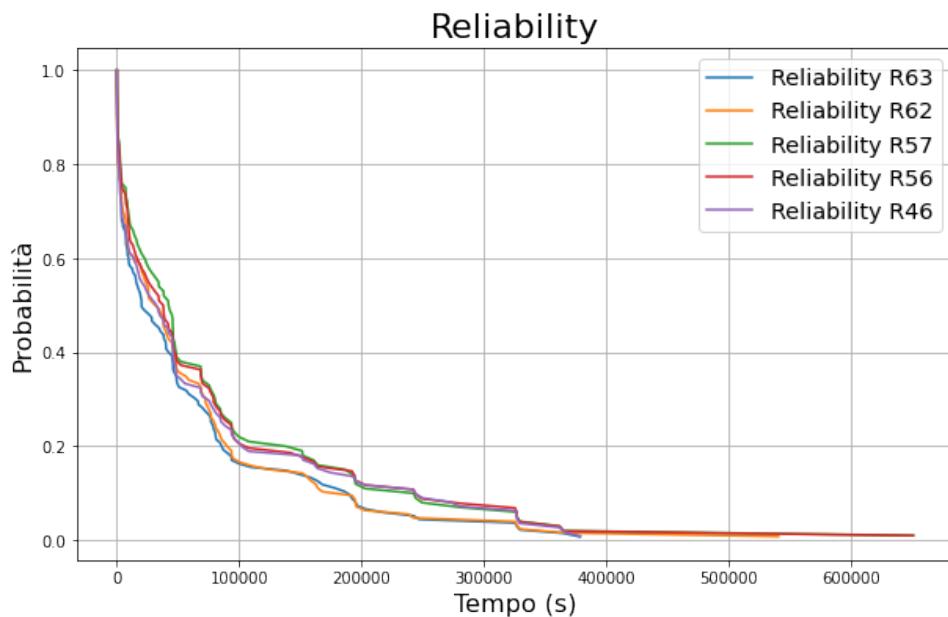


Figura 5.48: Reliability per rack critici

5.5.5 Analisi per nodi

La stessa tipologia di analisi è stata condotta anche per i nodi del sistema.

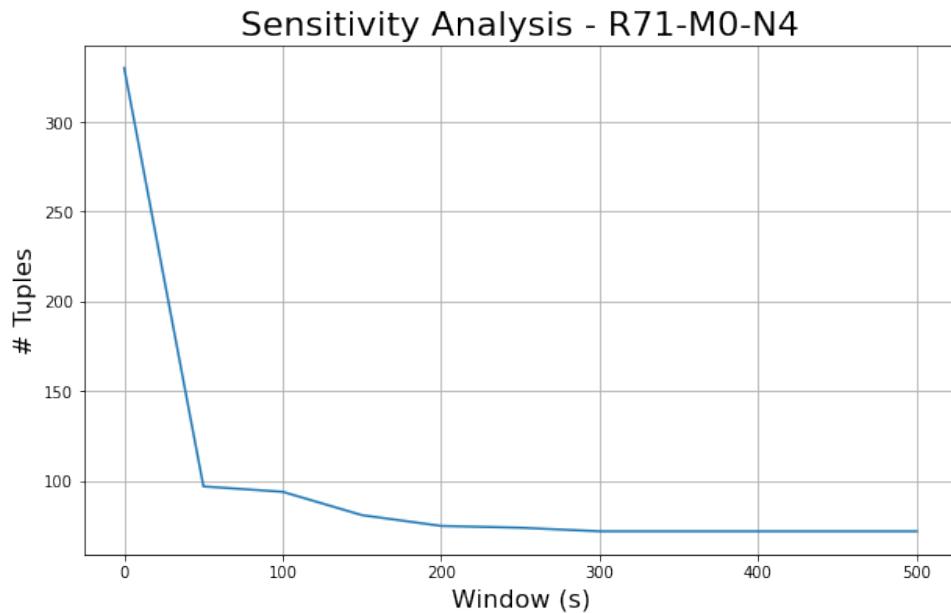


Figura 5.49: Sensitivity Analysis - R71-M0-N4

La finestra di coalescenza più appropriata è $W = 50$.

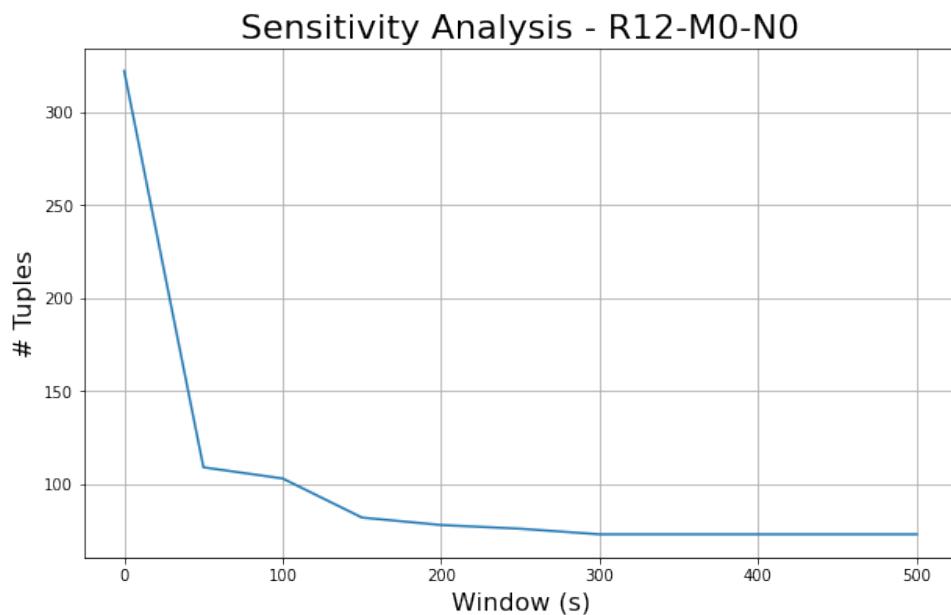


Figura 5.50: Sensitivity Analysis - R12-M0-N0

La finestra di coalescenza più appropriata è $W = 50$.

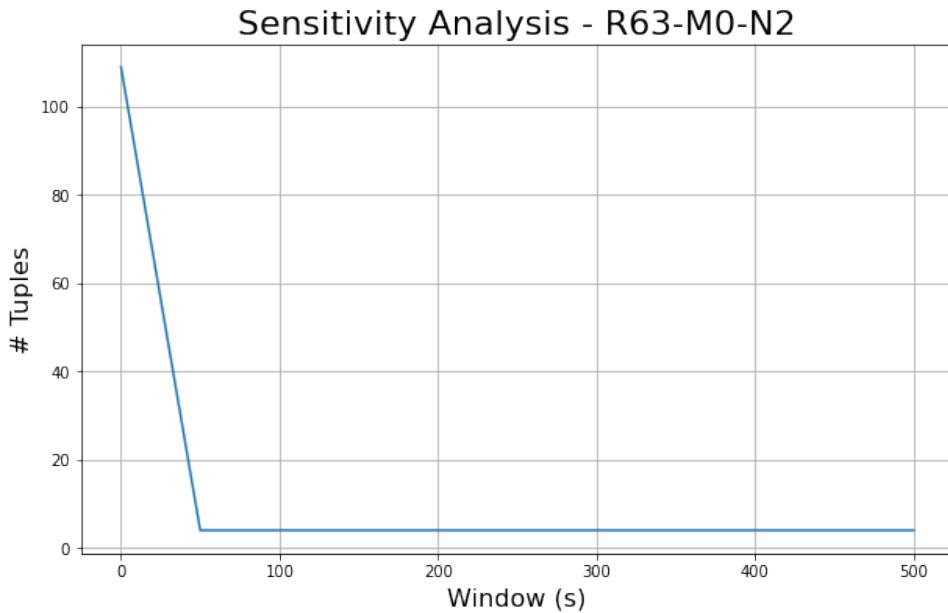


Figura 5.51: Sensitivity Analysis - R63-M0-N2

La finestra di coalescenza più appropriata è $W = 50$.

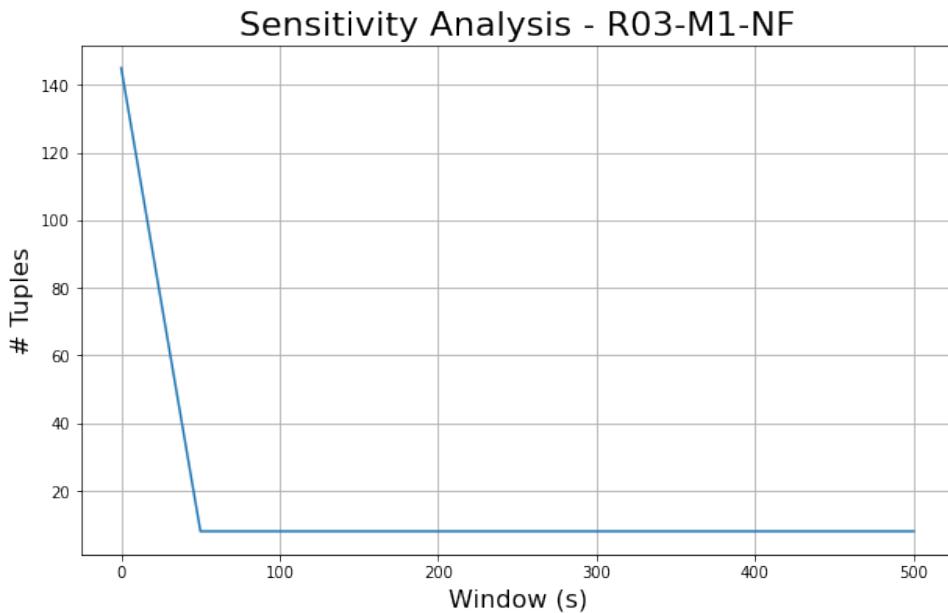


Figura 5.52: Sensitivity Analysis - R03-M1-NF

La finestra di coalescenza più appropriata è $W = 50$.

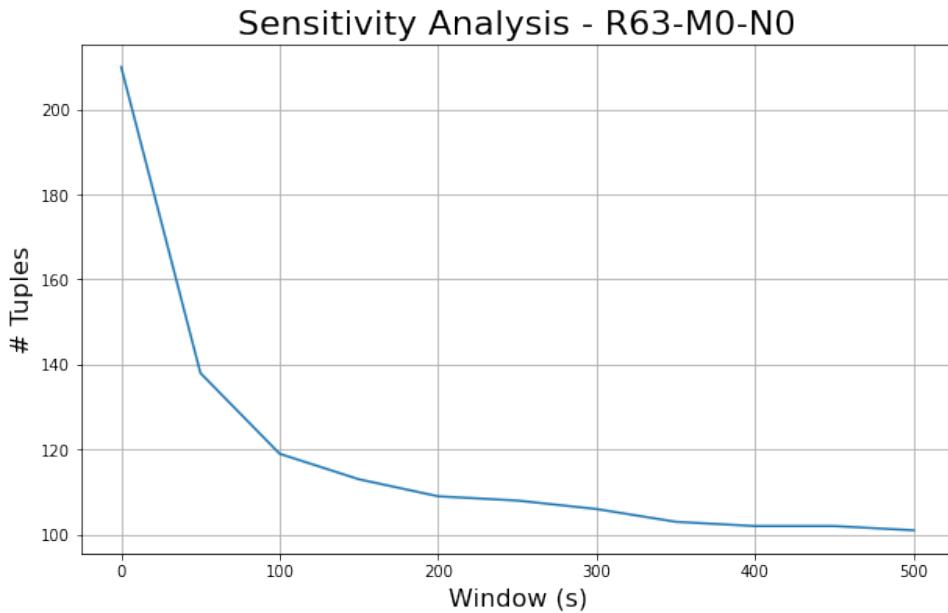


Figura 5.53: Sensitivity Analysis - R63-M0-N0

La finestra di coalescenza più appropriata è $W = 100$.

L'analisi condotta ha dunque portato ai seguenti risultati:

Nodo	W	# tuple
R71-M0-N4	50	97
R12-M0-N0	50	109
R63-M0-N2	50	4
R03-M1-NF	50	8
R63-M0-N0	100	119

Con le finestre di coalescenza riportate, è stata calcolata la reliability empirica di seguito riportata. I nodi *R71-M0-N4*, *R12-M0-N0*, *R63-M0-N0* sembrano essere reliable allo stesso modo.

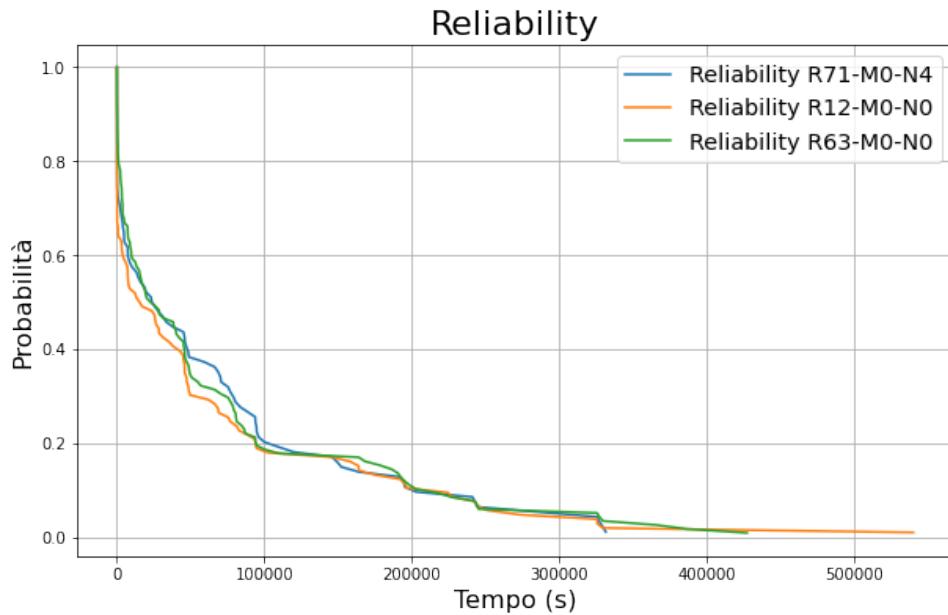


Figura 5.54: Reliability per nodi critici

5.5.6 Analisi per Card

La classica analisi è stata ripetuta anche per le 2 Card più critiche dell'architettura, ossia *J18-U01* e *J18-U11*, le quali come detto sono dedicate all'I-O del sistema.

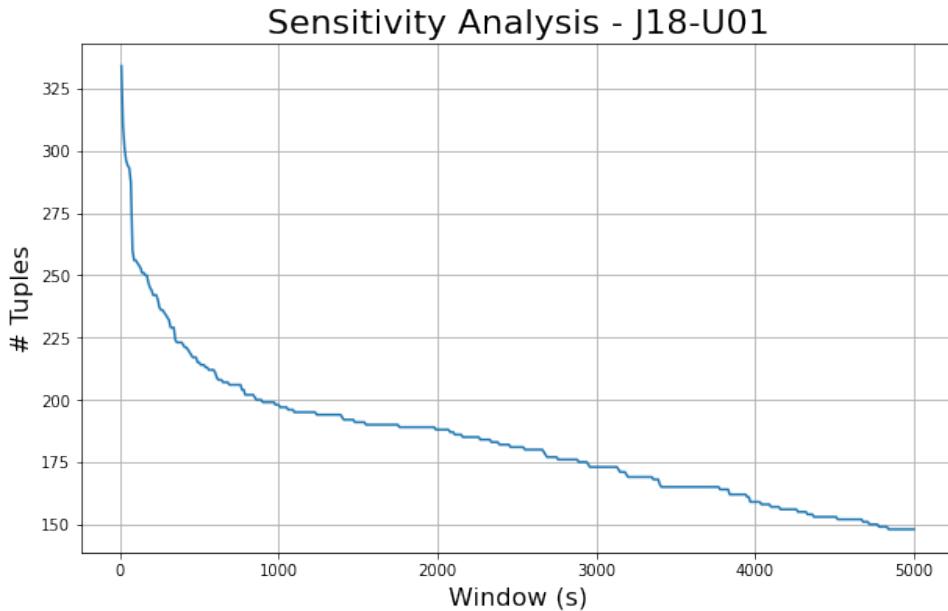


Figura 5.55: Sensitivity Analysis - *J18-U01*

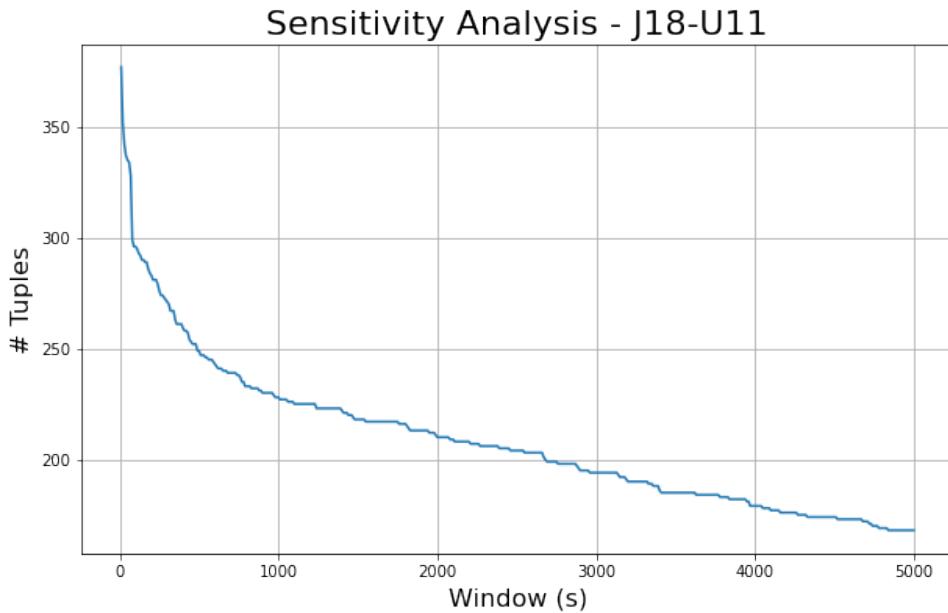


Figura 5.56: Sensitivity Analysis - *J18-U11*

I risultati pervenuti sono riassunti nella seguente tabella:

Card	W	# tuple
J18-U01	750	206
J18-U11	750	238

In ultima battuta, si è proceduto al calcolo della reliability empirica per entrambe le card, da cui ne risulta come J18-U11 è (di poco) la card meno reliable:

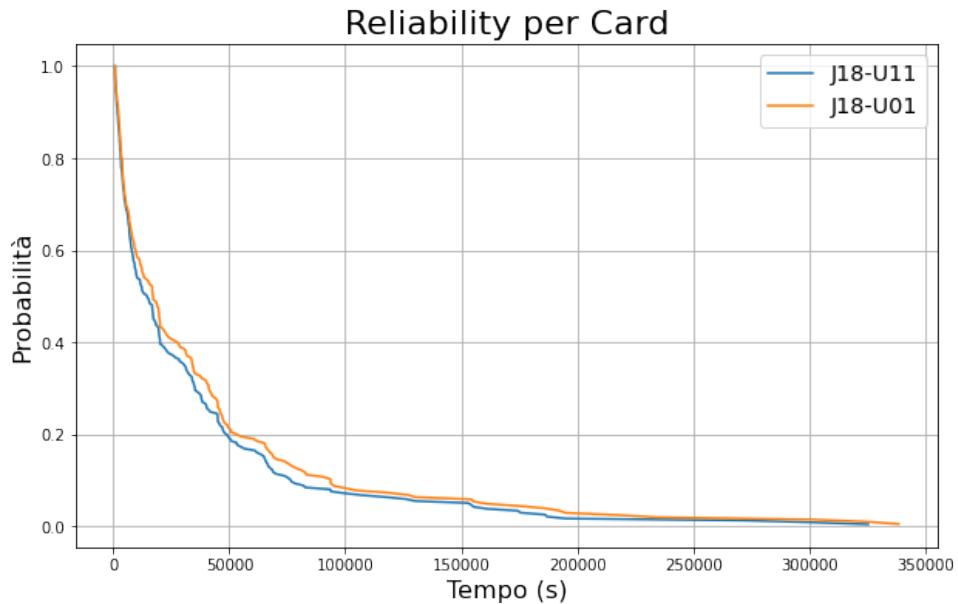


Figura 5.57: Reliability per Card

5.5.7 Analisi del *node daemon* CIOD

Dato che l'analisi del bottleneck ha evidenziato come il principale bottleneck del sistema possa essere l'intero sottosistema di I-O, si è ritenuto opportuno condurre un'analisi di esso, ricercando all'interno dei log tutti gli eventi registrati dal node daemon CIOD, relativo per l'appunto al monitoraggio del sottosistema di I-O.

La sensitivity analysis ha quindi condotto ai seguenti risultati:

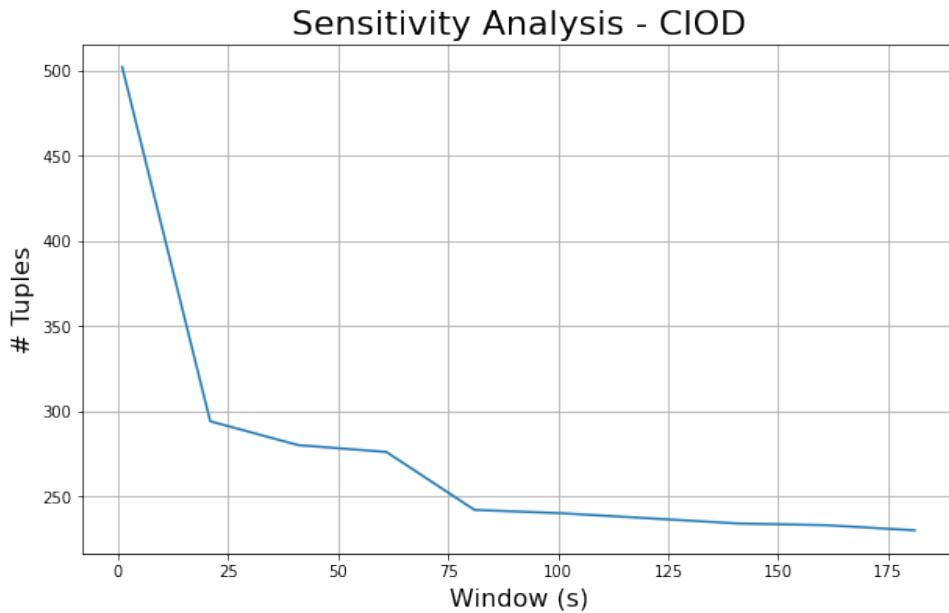


Figura 5.58: Sensitivity Analysis - I-O

Scegliendo una finestra di coalescenza $W = 21$, che comporta la generazione di 294 tuple, si è proceduto successivamente al tracciamento della reliability empirica dei fallimenti relativi al sottosistema di I-O a partire dalla CDF dei tempi di interarrivo:

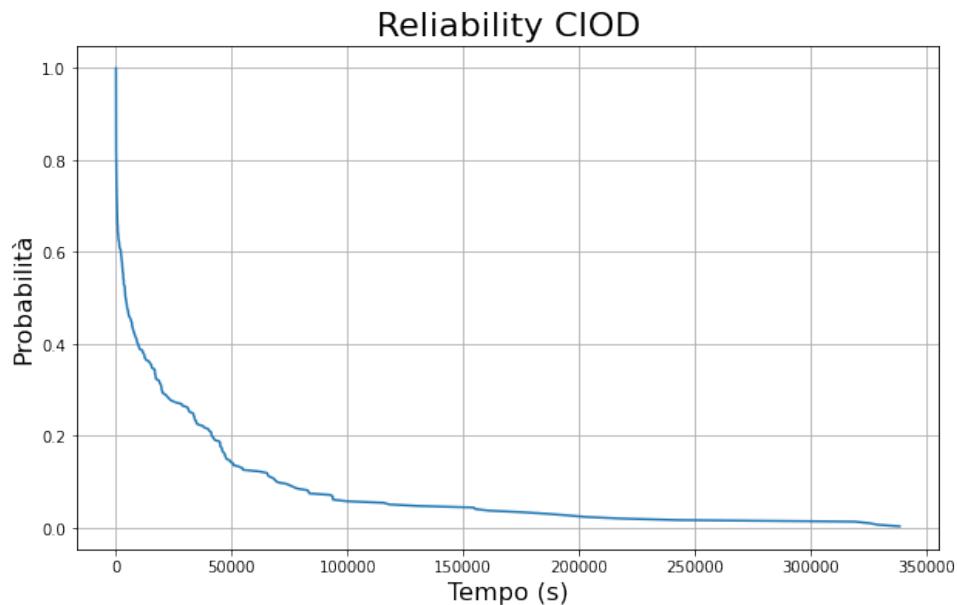


Figura 5.59: Reliability sottosistema di I-O

5.6 Confronto Mercury - BlueGene/L

Ultimate tutte le analisi sui due supercalcolatori, si conclude la trattazione con un confronto tra i risultati prodotti da esse. Partendo dai bottleneck dei sistemi, se i log di Mercury risultano essere totalmente polarizzati dal nodo computazionale *tg-c401*, al quale fanno riferimento più del 77% degli eventi totali e che rappresenta senza dubbio il bottleneck del sistema, lo stesso non può dirsi di Bluegene/L: nessun nodo del supercalcolatore può essere individuato come il principale collo di bottiglia del sistema. Tuttavia, come analizzato in precedenza, il bottleneck di BlueGene/L è probabilmente il sottosistema di *I/O*, in quanto oltre il 75 % dei fallimenti registrati dal sistema sono stati catturati dal demone CIOD (Console I-O Daemon). Proseguendo su tale ragionamento, risulta che gli errori registrati nei log di BlueGene/L sono distribuiti piuttosto equamente sia nel tempo che nello spazio, ossia nei vari nodi costituenti l'architettura del sistema: l'intervallo temporale in cui si concentrano più densamente i fallimenti è infatti sporadico e di breve durata (circa 48 minuti). Di contro, in Mercury la maggior parte degli eventi del log sono stati registrati in un intervallo specifiche di tuple, coprente un arco temporale di circa 3 giorni e contenente il 75% della totalità delle tuple; d'altronde, esso è anche l'intervallo in cui si registra il fallimento del nodo bottleneck, che ha influito pesantemente su tale stima. Da tali osservazioni si può intuire come con tutta probabilità il sistema BlueGene/L sia in realtà più reliable del sistema Mercury. Tale intuizione è confermata dal confronto diretto della reliability dei due sistemi:

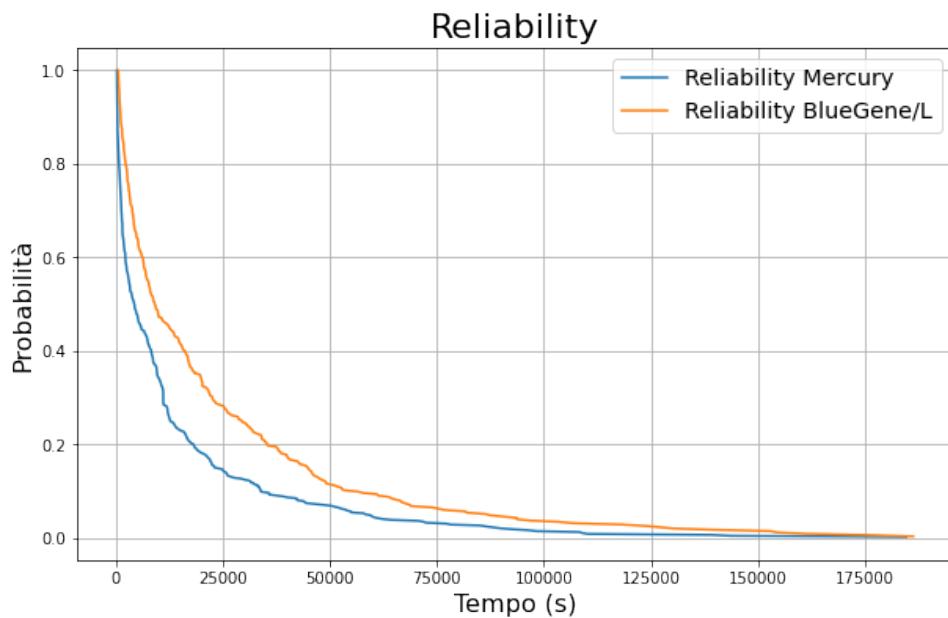


Figura 5.60: Reliability Mercury & BlueGene/L