UNIVERSITÀ DEGLI STUDI DI
NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea Magistrale in Ingegneria Informatica

Elaborato di

Impianti di Elaborazione

Anno Accademico 2021/2022

Professore
Domenico Cotroneo
Ingegneri
Pietro Liguori, Stefano Rosiello

Componenti del gruppo
Margherita Maria M63001118
Martina Russo M63001128
Michelle Pepe M63001196

Indice

Capitolo 1

Benchmarking

1.1 Traccia

Confrontare le prestazioni di due differenti sistemi (CPUs) sfruttando la libreria di benchmark **NBody**.

1.2 Strumenti Utilizzati

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

1.3 Introduzione

Nbody simula l'evoluzione di N corpi in un sistema gravitazionale: con questa simulazione vengono gestiti i corpi nello spazio, dove è preponderante la forza gravitazionale e l'interazione tra i corpi avviene secondo un sistema di equazioni a derivate parziali con cui analizzo il movimento dei corpi. La simulazione dà il numero di corpi e genera le matrici di movimento.
Si usa Nbody perché è uno dei benchmark più usati oggi perché guarda allo stack, al numero dei core e alle operazioni floating point.
L'obiettivo nostro è comparare due sistemi con stesso sistema operativo ma processori differenti.
Nel contesto dell'esercizio, sono stati scelti due sistemi con i seguenti processori:

- **AMD-Ryzen 7 37001**;
- **Intel(R) Core(TM) 10TH i7-10510U CPU @ 1.80GHz 2.30 GHz**.

Aumentando la dimensione N, quindi il numero di corpi, aumenta la complessità dell'algoritmo che è $O(N^2)$, e nel nostro elaborato ne abbiamo considerati 5000.

1.4 Soluzione

L'analisi del problema è partita con la creazione di un pre-campione contenente 30 misurazioni indipendenti di 4 osservazioni ciascuna. Per garantire che le osservazioni fossero indipendenti e identicamente distribuite, 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 pre-campione è stato necessario per poter calcolare la dimensione campionaria, fissando il valore di confidenza al 95% e l'errore pari a 152 micro-secondi. Una volta calcolata la dimensione campionaria, si è proceduto a ri-effettuare le misure dei campioni; in seguito abbiamo effettuato un T-Test paired 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 Nbody, possiamo essere certi dell'indipendenza delle varie osservazioni.

1.4.1 Stima della dimensione campionaria

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

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

Dove:
• E è l'upper bound dell'errore in time, tramite il quale effettiamo la stima della dimensione campionaria. Si è scelto E=152.
• σ è approssimato con la deviazione standard del precampione.
• $z_{\frac{\alpha}{2}}$ è il quantile calcolato dalla distribuzione normale impostando $\alpha = 0.05$; 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 massimo valore di dimensione campionaria tra i due sistemi, in modo da avere una stima significativa dei tempi per entrambi i processori. In figura 1.1 e 1.2 riportiamo i risultati e le dimensioni scelte:

Media Campione Intel:	Media Campione Ryzen:
2815,5	4674
1174	3337,25
1241,25	2663,5
1253,75	2679,25
1690,75	3219,25
1116	2657
2642,25	2774,25
1050	3433,5
1134,5	3021,25
1175	5605
2470,25	2691,5
1552	2903,5
1302	2652,25
1566,75	2645,25
1251	3118
1076,25	2948,25
1228	2743,25
1076,5	3008,25
1147,25	2741,5
1146,25	2646
1702	2651,25
1021,75	2817
1081	3493,75
1040,5	2690,5
1360,5	2679,25
1099,5	2832,25
1024,25	3069,5
2989,5	2679,5
1211	3325,25
1079	3098
Media Tot.	**Media Tot.**
1423,941667	3049,941667

Figura 1.1: **Tabella tempi medi pre-campione.**

Errore :	Deviazione Std. Intel:	Deviazione Std. Ryzen:	Dim. Campionaria Intel	Dim. Campionaria Ryzen	Dimensione Scelta
152,4970833	556,936613	635,8006447	51,23899542	66,77762913	67

Figura 1.2: **Tabella dimensioni campionarie.**

Di conseguenza, come si evince dalla figura 1.2, per le considerazioni fatte in precedenza, si è scelto N=67.

1.5 Generazione campioni differenza

A questo punto, è stato rieseguito un test, con un numero di osservazioni pari a 67.
Di seguito sono riportati i campioni differenza , calcolati sottraendo coppie di

osservazioni per i due sistemi scelti:

Intel	Ryzen	Differenza
1218	2127	909
1499	2117	618
1315	1821	506
1278	2868	1590
1266	1900	634
1075	1827	752
1027	1910	883
1090	1858	768
1058	1905	847
1034	1884	850
1064	2063	999
1094	2016	922
1075	1899	824
1098	1902	804
1063	1854	791
1057	1938	881
1037	1928	891
1036	2479	1443
1046	1891	845
1035	2024	989
1172	2307	1135
1080	1900	820
1065	2707	1642
1053	1881	828
1202	2600	1398
1038	1943	905
1050	1935	885
1082	2263	1181
1032	2074	1042
1036	1975	939
1032	1914	882
1046	2134	1088
1050	1865	815
1117	1929	812
1050	2044	994
1088	1961	873
1049	1952	903
1076	1876	800

Figura 1.3: **Campioni differenze per N=5000.**

1.6 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 sul campione differenza. Di seguito sono riportati i risultati.

1.6.1 Analisi N=5000

Per quanto riguarda la size 5000 sono stati ottenuti i seguenti risultati:

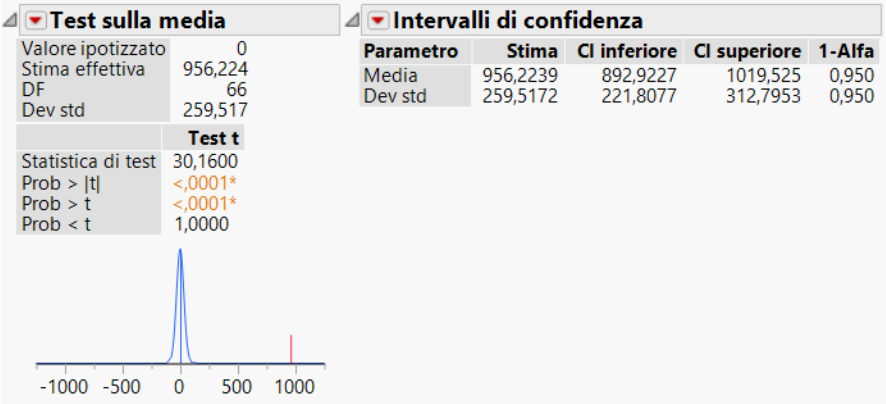


Figura 1.4: **T-Test N=5000.**

La differenza media tra il dispositivo dotato di processore Intel e quello dotato di processore Ryzen è di 956,224 ms.
Media e mediana a valori vicini confermano che orientativamente al crescere del numero dei campioni la distribuzione tenda a una normale, lasciando intendere che le ipotesi di campioni indipendenti e identicamente distribuiti siano effettivamente valide.
Come si evince dalla figura seguente, il test è stato rigettato.

1.7 Considerazioni

Indipendentemente dal campione differenza analizzato, è possibile concludere che:
• Si può confermare la validità del teorema, e quindi l'indipendenza dei campioni, utilizzando come indicatore la similarità tra i valori di media e mediana. In particolare, nell'esempio analizzato abbiamo una differenza tra i due pari a circa 70.

• Effettuando il test in cui è specificata l'ipotesi nulla H0 : μ = 0 e un valore di soglia α= 0.05, si può procedere all'analisi dei risultati restituiti da JMP, in particolare del p-value riportato sotto la voce P $\leqslant$ 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. Nel nostro caso viene rigettata l'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%. Il risultato allora ci dice che possiamo concludere che il processore Intel i7 10TH risulta essere più veloce dell'AMD-Ryzen.

• un ulteriore conferma dell'analisi svolta può essere ottenuta tramite l'utilizzo degli intervalli di confidenza: notiamo come nell'intervallo definito, non è compreso lo 0, quindi si è confidenti al 95% che non sussiste l'ipotesi nulla e quindi c'è una differenza statistica dovuta a un fenomeno e non a una casualità.

Capitolo 2

PCA e Clustering

2.1 Traccia

Dai dati di un workload reale, travare un workload sintetico che segua la seguente proprietà: gli studenti trovino il miglior trade-off tra numero di componenti principali e numero di cluster.

2.2 Filtering

Il dataset di partenza è composto da 24 colonne e 3000 righe. Ogni colonna corrisponde ad un parametro che è stato ottenuto monitorando le risorse del sistema.
La colonna **Slab** presenta tutti i valori pari a 0, tranne riga 90 e riga 510, le quali sono state eliminate in quanto a seguito di un'operazione di clustering emergevano cluster fatti da un solo elemento e da ciò si e dedotto che rappresentassero degli outliers.

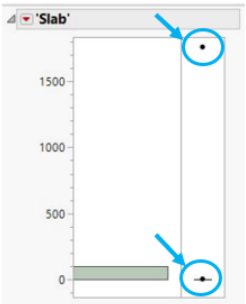


Figura 2.1: **Colonna Slab.**

Tra le 24 colonne sono state cancellate quelle che presentavano valori costanti ovvero: **Active, Slab, AvgLatency, Errors, ANonpages**.

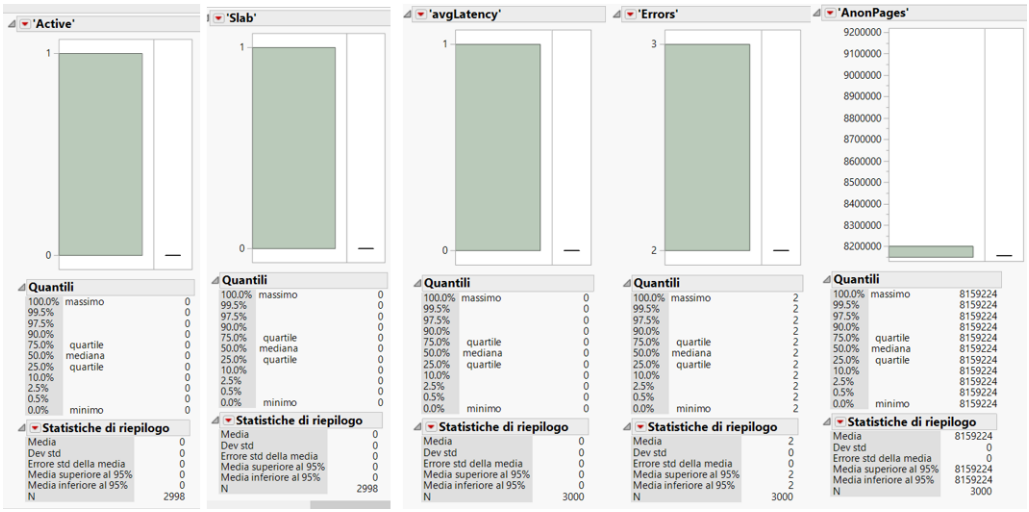


Figura 2.2: **Colonne eliminate.**

Un'ulteriore considerazione è stata fatta riguardo gli outliers mostrati in figura 2.3. Essi corrispondono alla stessa riga, in particolare la prima riga del dataset. E' stato visto che il carico in questo caso è basso, e per ridurre la varianza da spiegare, è stato deciso di escludere l'intera riga dal dataset.

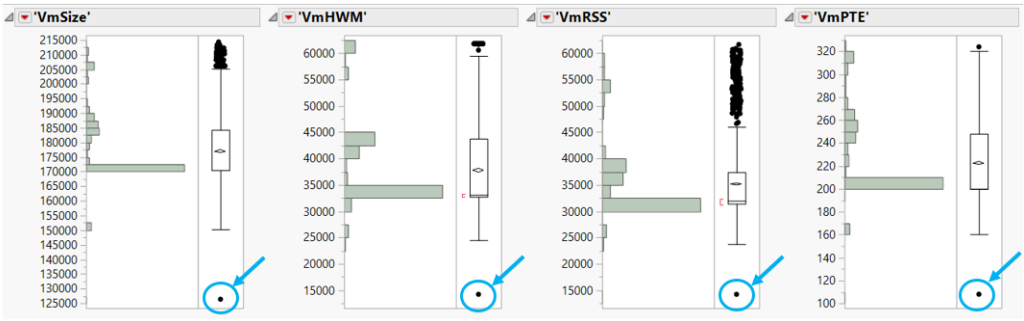


Figura 2.3: **Outliers riferiti alla prima riga.**

Un'ultima considerazione sugli outlier è stata fatta guardando il box plot di MemFree, figura 2.4: sono presenti alcuni outliers che corrispondono alle prime 84 righe del dataset. In corrispondenza di queste ultime possiamo notare valori significativi della componente WriteBack: abbiamo giustificato questo fenomeno

associando queste righe alla fase di accensione del sistema, caratterizzata da numerosi cache miss.

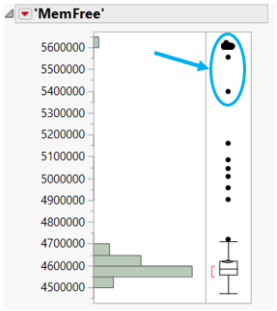


Figura 2.4: **Box plot MemFree.**

Per tale motivo abbiamo considerato queste righe come una parte transitoria del sistema. Il transitorio di un sistema solitamente non viene preso in considerazione ai fini dello studio di esso, a meno che non rappresenti un sistema cloud. In questo caso non sapendo la natura del sistema, abbiamo preferito non omettere tale fase.

2.3 PCA

La PCA è una tecnica di riduzione della dimensionalità del problema. In particolare, date le variabili del workload, l'idea di base è di individuare nuove componenti indipendenti tra loro (quindi in-correlate) attraverso una trasformazione di stato e ordinandole da quella che esprime più variabilità a quella che esprime meno variabilità.
Questa tecnica non fa altro che **una trasformazione dello spazio di stato**. Prende i parametri del workload e li riporta in uno stato in cui la matrice che genera tale stato è una matrice di correlazione. I nuovi componenti sono incorrelati e li posso ordinare secondo un criterio di varianza spiegata.
La trasformazione di stato tra x e y mi deve portare a dei fattori che sono:
- Linearmente indipendenti.
- I nuovi punti $y_1, y_2, \ldots, y_n$ devono essere ordinati secondo la varianza spiegata.

Se ho una varianza iniziale da dei punti dello spazio, se cambio lo spazio, la varianza non cambia. Nel nuovo dataset trasformato, poiché Y sono incorrelati, la varianza è allocata sui vari $y_1, y_2, \ldots, y_n$ in maniera decrescente.
Abbiamo detto che la PCA è una trasformazione di stato e per farlo devo trovare gli autovalori della mia matrice e costruire auto-vettori.
Riportiamo a tal proposito in figura 2.5 la matrice degli autovalori: la nostra trattazione prevede di vedere la differenza nell'utilizzo di 3, 4 o 5 componenti.

Autovalori

Numero	Autovalore	Percentuale	20 40 60 80	Percentuale cumulativa
1	10,6175	55,882		55,882
2	3,6786	19,361		75,243
3	2,1374	11,249		86,859
4	0,8469	4,457		91,086
5	0,6351	3,343		94,583
6	0,3566	1,877		96,168
7	0,2983	1,570		97,738
8	0,1873	0,986		98,724
9	0,1057	0,557		99,281
10	0,0910	0,479		99,759
11	0,0178	0,094		99,853
12	0,0114	0,060		99,913
13	0,0078	0,041		99,954
14	0,0058	0,030		99,985
15	0,0014	0,007		99,992
16	0,0009	0,005		99,997
17	0,0006	0,003		100,000
18	0,0000	0,000		100,000
19	0,0000	0,000		100,000

Figura 2.5: **Matrice degli autovalori.**

2.3.1 PCA e clustering per 3 componenti

Se scegliamo 3 componenti, abbiamo una varianza di : **86.86%**.
- 30 cluster -> quanta varianza prendiamo? **86.46%**.
- 70 cluster -> quanta varianza prendiamo? **86,73%**.
- 80 cluster -> quanta varianza prendiamo ? **86,75%**.
- 90 cluster -> quanta varianza prendiamo? **86,77%**.
Di seguito riportiamo il Dendogramma, il quale, attraverso l'approccio visivo, ci consente di capire quanta varianza perdiamo o meno al variare del numero di cluster:

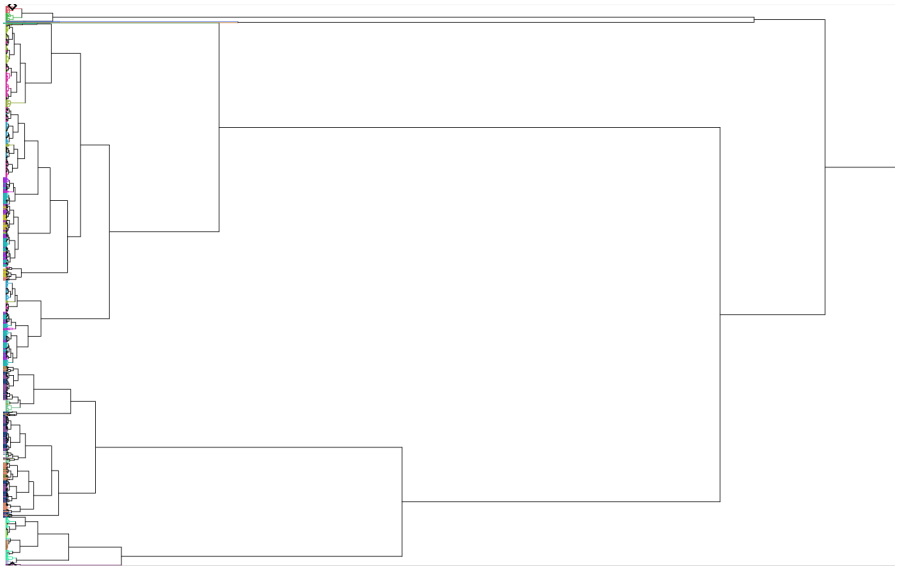


Figura 2.6: **Dendogramma per 3 componenti principali.**

2.3.2 PCA e clustering per 4 componenti

Se scegliamo 4 componenti, abbiamo una varianza di : **91,09%**.
- 30 cluster -> quanta varianza prendiamo? **89.96%**.
- 70 cluster -> quanta varianza prendiamo? **90,75%**.
- 80 cluster -> quanta varianza prendiamo? **90,81%**.
- 90 cluster -> quanta varianza prendiamo? **90,84%**.

Di seguito riportiamo il Dendogramma:

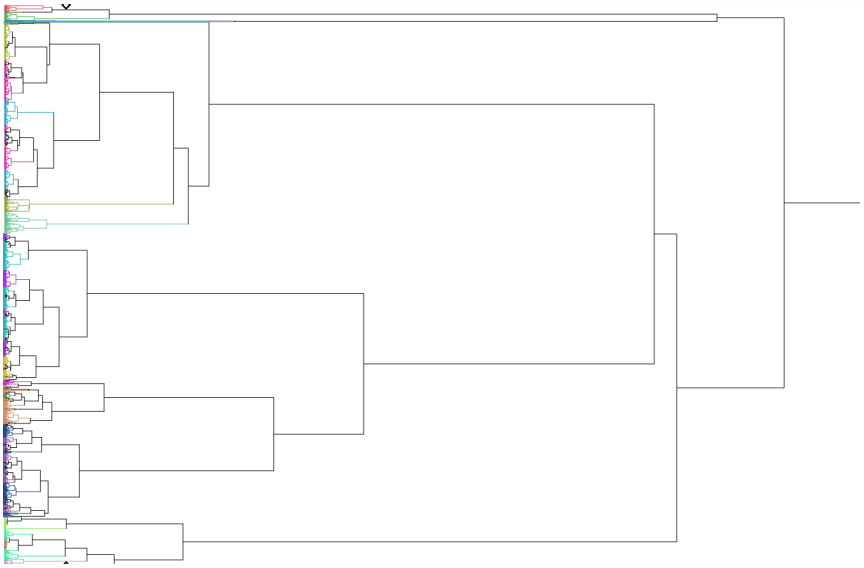


Figura 2.7: **Dendogramma per 4 componenti principali.**

2.3.3 PCA e clustering per 5 componenti

Se scegliamo 5 componenti, abbiamo una varianza di : **94,58%**.
- 30 cluster -> quanta varianza prendiamo? **92,36%**.
- 70 cluster -> quanta varianza prendiamo? **93,78%**.
- 80 cluster -> quanta varianza prendiamo? **93,88%**.
- 90 cluster -> quanta varianza prendiamo? **93,99%**.

Di seguito riportiamo il Dendogramma:

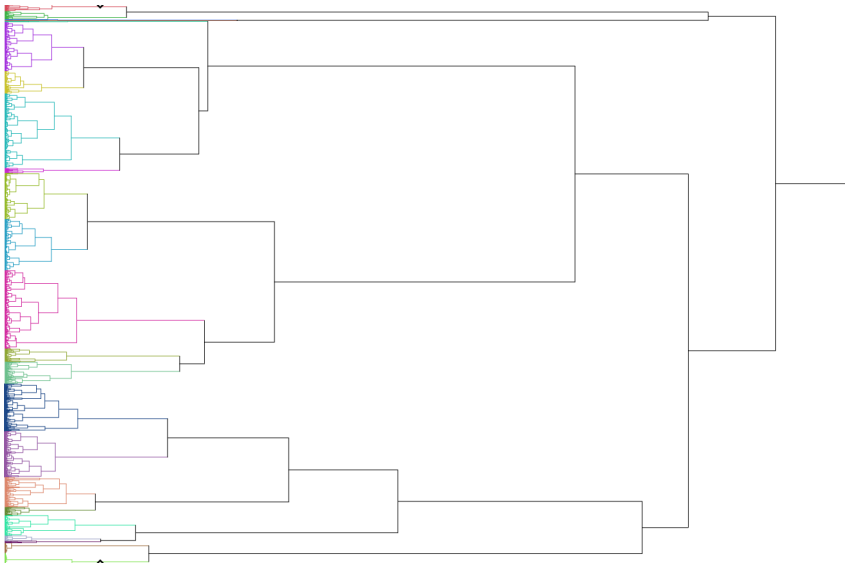


Figura 2.8: **Dendogramma per 5 componenti principali.**

2.4 Conclusioni

Riportiamo una tabella che sintetizza la varianza spiegata nei vari casi:

Cluster	3 PC	4 PC	5 PC
30	86,46	89,96	92,36
70	86,73	90,75	93,78
80	86,75	90,81	93,88
90	86,77	90,84	93,99

Figura 2.9: **Percentuali varianza spiegata.**

Cluster	3 PC	4 PC	5 PC
30	13,54	10,04	7,64
70	13,27	9,25	6,22
80	13,25	9,19	6,12
90	13,23	9,16	6,02

Figura 2.10: **Percentuali varianza persa.**

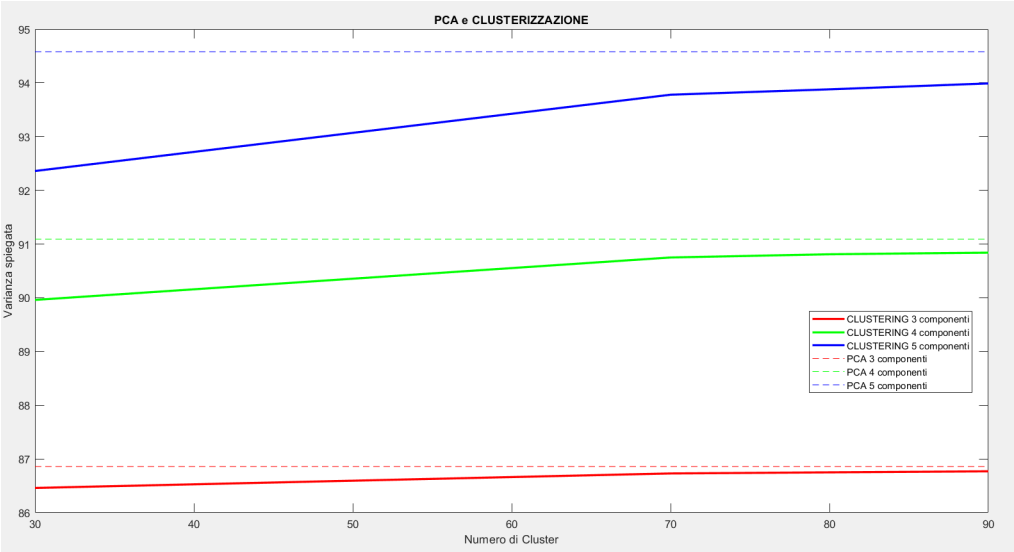


Figura 2.11: **PCA e Clusterizzazione.**

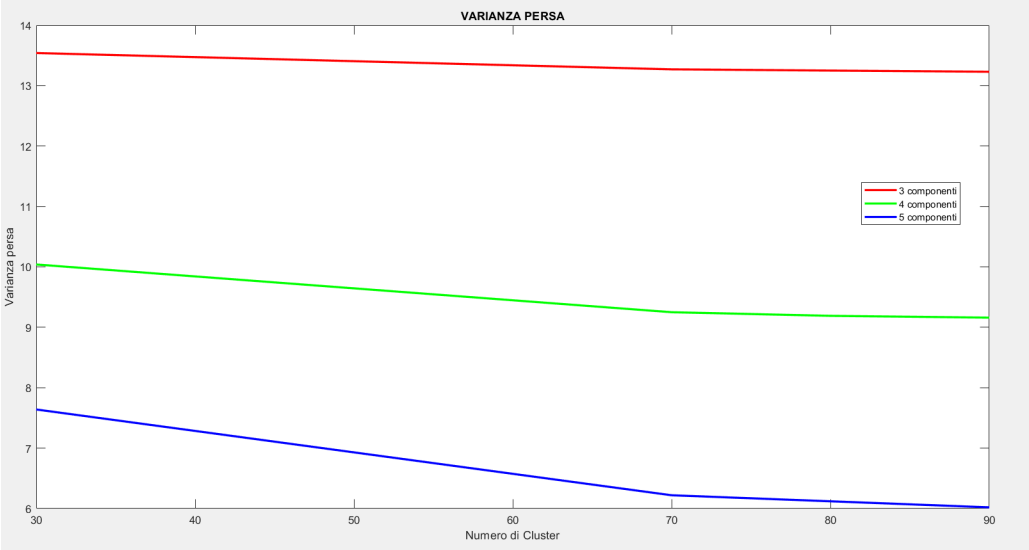


Figura 2.12: **Varianza persa.**

Osservando i grafici in alto, abbiamo scelto per il nostro workload sintetico, di considerare 4 componenti e 70 cluster, per differenti motivi:
1) la distanza tra varianza spiegata solo con PCA e varianza spiegata con Clustering, risulta essere minima nel caso di 3 componenti principali, ma rispetto a

considerare 4 componenti principali, guadagniamo decisamente una percentuale maggiore;
2) il punto sul grafico (x,y)=(70,4), risulta essere il primo punto dopo il quale la varianza inizia ad assestarsi, per cui considerare un numero maggiore di cluster non porterebbe nessun vantaggio considerevole.

Per definire il workload sintetico è stato scritto uno script in Python, che per ogni cluster seleziona in modo casuale una riga: in figura 2.13 riportiamo il workload sintetico così determinato.

	'VmPeak'	'VmSize'	'VmHWM'	'VmRSS'	'VmPTE'	'Threads'	'MemFree'	'Buffers'	'Cached'	'Inactive'	'Dirty'	'Writeback'	'Mapped'	'PageTabl'	'Committ'	'NumOfA'	'proc-fd'	'avgThroughpu'	'avgElapsed'
32	152272	150216	25172	25104	160	9	5606344	32268	334648	141316	302968	5606344	56	77860	23772	27716	7192	294908	510
59	153484	151436	26612	26468	160	9	5601880	34896	334676	142716	305592	5601880	12	78736	23772	27972	7192	296532	1020
80	153484	151436	26612	26468	160	9	5554656	55856	342108	149612	327972	5554656	3944	79596	23976	45512	7436	299332	1020
81	153484	151436	26612	26468	160	55	5397504	81168	435176	167336	430616	5397504	4764	81704	24376	81332	7424	302560	1530
9	152272	150216	24628	24576	160	10	5609436	30020	334636	140700	300792	5609436	132	76736	23708	27800	7192	297204	1530
15	152272	150216	24696	23824	160	9	5609576	30596	334640	140024	301308	5609576	140	76080	23772	27788	7196	294504	2040
82	153484	151436	26612	26468	160	45	5160008	83244	656460	230156	597448	5160008	221152	87804	25312	89796	7796	304888	2040
83	153484	151436	26612	26468	160	32	5084564	83540	693480	327380	575804	5084564	121820	126116	28324	90284	7984	343200	2040
87	170344	170340	31144	31140	200	64	4902916	84164	864428	454384	624920	4902916	142552	130788	28324	95252	8024	366024	2040
88	170344	170340	31144	31140	200	43	4471580	94884	1259568	597900	891332	4471580	459632	134504	29956	116216	8024	365692	1530
89	170344	170340	31148	31140	200	26	4720436	97808	1064968	366496	880988	4720436	192588	84604	24312	109564	7576	315392	1530
90	170344	170340	31208	31140	200	13	4709868	97944	1075684	376648	882056	4709868	119972	85176	24272	109568	7704	317992	1530
91	170344	170340	31916	31912	200	9	4699488	98052	1084804	386696	882460	4699488	107284	86152	24092	109392	7620	319664	1020
92	170344	170340	31920	31912	200	37	4689572	98160	1094124	395720	882812	4689572	77100	86132	24168	109424	7608	317620	1530
93	170344	170340	31920	31912	200	51	4673128	98276	1110480	406568	888336	4673128	57796	86164	23948	109840	7320	319164	1530
94	170344	170340	31920	31912	200	45	4667772	98440	1117028	407616	892048	4667772	43668	84208	23740	109340	7208	317168	1530
1126	172392	170340	32868	31920	200	35	4598556	166948	1117468	435456	933228	4598556	32	84264	23844	110472	7208	315436	1530
1812	172392	170084	33228	32116	200	29	4574496	190124	1117892	465116	927360	4574496	188	84460	23844	111020	7208	318036	1530
1578	172392	170340	33228	31568	200	64	4582144	182812	1117772	454564	929928	4582144	152	83908	23844	110912	7204	316256	1530
1859	176424	173608	34560	34080	208	111	4568400	191124	1117916	468912	926544	4568400	144	86392	23844	114568	7212	320944	1530
478	172388	170340	32696	31460	200	58	4636060	131236	1117180	412004	920212	4636060	140	83800	23740	109468	7208	318256	1530
917	172392	170340	32696	31520	200	60	4606908	159496	1117376	425612	935120	4606908	32	83860	23740	110180	7204	316148	1530
167	172388	170340	32192	31916	200	19	4663416	104284	1117060	408900	896660	4663416	40	84216	23740	108944	7208	317788	1530
361	172388	170340	32696	30768	200	36	4648528	120072	1117140	410956	909356	4648528	164	83100	23740	109236	7208	314464	1530
397	172388	170340	32696	31260	200	9	4644528	123540	1117152	411536	912748	4644528	40	83592	23740	109256	7208	318872	1530
855	172392	170340	32696	31480	200	26	4611884	155488	1117348	423764	932892	4611884	164	83880	23740	110084	7204	318824	1530
261	172388	170340	32696	32168	200	39	4656420	111172	1117096	411480	901284	4656420	164	84496	23740	108940	7208	318292	2040
1087	172392	170340	32696	31996	200	33	4601920	164876	1117444	433880	932780	4601920	152	84200	23844	110380	7208	314012	2040
1991	185964	182116	41280	38868	236	83	4561484	193848	1117964	479004	924024	4561484	140	90936	23848	113536	7244	327384	2040
1962	185340	180144	41280	38328	228	108	4561560	193196	1117960	477364	924436	4561560	164	90692	23848	114564	7232	329428	1530
2293	192256	183208	43804	37056	248	106	4556976	198772	1118136	488256	918048	4556976	24	89144	23848	113900	7032	332340	1530
2129	189652	183360	41968	38016	244	83	4559016	196712	1118060	483888	921236	4559016	32	90572	23848	113744	7248	326852	1530
2489	192384	186160	43804	37960	256	120	4553364	201780	1118232	496576	913684	4553364	24	90060	23848	113848	7264	332924	1530
2718	197292	189740	43804	37816	272	107	4551876	203500	1118344	502388	909608	4551876	56	90168	23848	113756	7280	340108	1530
143	172388	170340	32192	31912	200	57	4664840	102560	1117044	408280	895540	4664840	152	84216	23740	108868	7212	315884	1020
655	172392	170340	32696	30768	200	9	4626412	141752	1117264	417624	924504	4626412	32	83204	23740	109808	7208	314616	1020
817	172392	170340	32696	31372	200	45	4614840	152548	1117324	422876	930708	4614840	32	83548	23740	110084	7208	317992	1020
1008	172392	170340	32696	32028	200	20	4603256	162548	1117412	430408	933924	4603256	28	84368	23740	110316	7204	315764	1020
1429	172392	170340	32868	32120	200	9	4586932	178092	1117704	448664	931596	4586932	24	84464	23844	110692	7208	317052	1020
1372	172392	170340	32868	31628	200	47	4588612	176948	1117680	445816	932784	4588612	28	83972	23844	110652	7204	318504	1020
1858	176424	173608	34560	34084	212	105	4569544	191080	1117916	468868	926536	4569544	164	86348	23844	113040	7220	320800	1020
623	172388	170340	32696	31508	200	9	4627416	140272	1117248	416936	924436	4627416	24	83852	23740	109688	7204	315816	510
1881	178212	172368	36480	33492	208	69	4569616	191756	1117928	469160	926352	4569616	140	85996	23844	113104	7216	320872	510
1923	183580	176004	39436	36328	216	111	4564512	192520	1117940	473776	925352	4564512	140	88668	23848	114240	7220	321260	510
2362	192256	186300	43804	39104	256	78	4554552	199424	1118168	492664	916368	4554552	44	91448	23848	113952	7208	331512	0
2077	186864	181996	41320	35636	244	95	4562380	195560	1118028	479500	922064	4562380	12	87980	23848	113944	7252	328544	510
2284	192256	186900	43804	39472	256	127	4551464	198664	1118136	490380	918232	4551464	188	91984	23848	117708	7260	330828	510
2328	192256	184224	43804	36944	248	120	4557080	199136	1118152	489356	917212	4557080	52	89448	23848	113412	7032	331772	510
2677	197292	190856	43804	38696	272	124	4550860	203128	1118324	502096	910388	4550860	12	90792	23848	113512	7280	337248	510
2332	192256	183240	43804	37032	248	54	4557020	199156	1118156	489596	917088	4557020	24	89376	23848	113960	7472	331072	510
1984	185964	178076	41280	34160	224	82	4566116	193708	1117968	474012	924164	4566116	56	86496	23848	113700	7232	323572	1020
2025	186284	179520	41320	35408	228	47	4564248	194472	1117996	477108	923100	4564248	144	87896	23848	113760	7236	323988	1020
2202	189652	181852	41968	36792	240	126	4559036	197680	1118088	485092	919812	4559036	44	89168	23848	113776	7248	326744	1020
2495	192384	185744	43804	37300	256	117	4554028	201868	1118232	496148	913588	4554028	144	89704	23848	113680	7264	334456	1020
2281	192256	186836	43804	39908	256	111	4555328	198616	1118128	490748	918240	4555328	20	92248	23848	113276	7044	334872	1020
2322	192256	185564	43804	37756	252	75	4556064	199032	1118148	490020	917260	4556064	44	90320	23848	113684	7260	332444	1020
2536	192756	187772	43804	38628	264	106	4552496	202204	1118252	498572	912852	4552496	172	90968	23848	113852	7272	335632	1020
2554	192756	186656	43804	37888	260	72	4553396	202304	1118260	498192	912600	4553396	60	90292	23848	113908	7268	333540	1020
2729	197292	196920	46860	46860	284	160	4543500	203616	1118348	511596	909548	4543500	156	99200	23848	113024	7292	341592	1020
2311	192256	186212	43804	38700	256	102	4553876	198944	1118148	490604	917508	4553876	8	91108	23848	115344	7476	332480	1020
2735	202040	201016	49892	49888	296	161	4539556	203640	1118352	514720	909492	4539556	20	102228	23848	112688	7304	342532	1530
2966	215352	206136	61784	53204	312	9	4538724	204608	1118488	521028	907592	4538724	144	105548	23848	110172	7320	346964	1020
2993	215352	206136	61784	53204	312	9	4539220	204712	1118500	521348	907380	4539220	36	105544	23848	110156	7320	345724	1530
2786	208184	202040	55500	51292	300	160	4538692	203888	1118376	516780	909108	4538692	40	103628	23848	112744	7308	343524	1020
2802	209208	207160	56080	55640	312	161	4533964	203972	1118384	521348	908952	4533964	12	107952	23848	112696	7320	349364	1020
2835	215352	210232	61784	58536	312	160	4530532	204148	1118392	524704	908704	4530532	40	110872	23848	112836	7320	348412	1020
2791	208184	204088	55500	52520	304	161	4536700	203924	1118376	518084	909072	4536700	132	104860	23848	112892	7312	345288	510
2866	215352	209208	61784	56768	312	160	4532548	204288	1118444	523512	908332	4532548	28	109112	23848	112764	7316	349120	510
2840	215352	211256	61784	58928	312	161	4529004	204152	1118400	526000	908648	4529004	176	112112	24004	112904	7648	354464	1020
2846	215352	211256	61784	59796	320	161	4526384	204180	1118428	527616	908548	4526384	164	113552	23996	113304	7776	356100	510

Figura 2.13: **Workload sintetico.**

Capitolo 3

Web Server

3.1 Traccia

Analizzare le performance di un web server, effettuando le seguenti attività:
• **Capacity Test**: valutazione delle performance del sistema al variare del carico di lavoro imposto;
• **Workload Characterization**: estrapolare un workload sintetico a partire da un workload reale e verificarne la significatività statistica dei rispettivi workload di basso livello;
• **Design of Experiment**: verifica dell'influenza 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 JMeter, script in Python sviluppati nell'ambiente Visual Studio Code e script in Matlab necessari per effettuare grafici o validazioni di ipotesi (i codici di questi ultimi sono inclusi nelle cartelle corrispondenti, ma non vengono riportati in tale elaborato per brevità). Per il Design of Experiment si è invece fatto uso del software JMP.

3.3 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.

L'elemento principale di un test è il **Thread Group**: per "thread" intendiamo gli utenti virtuali che noi settiamo per fare le richieste alle risorse del server. Quindi, possiamo settare:
- **il numero di utenti che fanno le richieste**: lo abbiamo settato pari a 50;
- il **ramp-up period**, ovvero il tempo di attivazione dell'ultimo thread: lo abbiamo settato pari a 50 (di conseguenza abbiamo che si attiva un utente al secondo);
- il **loop count**, ovvero numero di volte che ogni utente deve essere attivato (abbiamo settato la durata di un test pari a 5 minuti).

L'elemento successivo necessario a tale elaborato è il **Sampler**, ovvero il campionatore. A noi interessano gli http Request, in quanto faremo richieste http. Nelle richieste http, andremo a specificare una serie di informazioni:
- **l'ip del server** (e dobbiamo mettere l'indirizzo IP del server creato con virtual box) (192.168.56.103);
- **il path**, che identifica il nome della risorsa (se la risorsa si trova in /var/www/html dobbiamo mettere solo il nome della risorsa perché di default cerca in quella cartella);
- **settiamo GET**, perché facciamo delle richieste;
- **port number 80**.

Ciò che ci serve per rendere il nostro piano di test più realistico, è il controllore: usiamo il **Logic Controller**, ed in particolare il **Random Controller**, che ci permette di aggiungere variabilità nelle nostre richieste.

L'altro elemento fondamentale è il Timer: **Costant Throughput Timer**. Questo mi permette di specificare il numero di richieste al secondo o al minuto, che ogni singolo thread o tutti i thread, devono fare: mi da il "rate di richieste". Nel nostro elaborato abbiamo inserito un CTT che varia in un intervallo tra 6200 (primo test effettuato) e 25000 (ultimo test effettuato).

L'ultimo elemento nel piano di test, è il **Listener**: ovvero colui che ci da l'output nel nostro piano di test. I listeners ci permettono di monitorare ciò che sta accadendo: abbiamo utilizzato il **Simple Data Writer**, che è un tipo di Listener che permette di specificare il nome del file (.csv) e di salvare informazioni per ogni singola richiesta.

Per ogni test sono state effettuate 3 ripetizioni e per ogni ripetizione, attraverso l'utilizzo di uno script in Python, sono stati calcolati il throughtput 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 3 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 thoughput (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 thoughput 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 thoughput in genere si riduce drasticamente.

Il test è stato effettuato su NoteBook Dell dotato di processore Intel I7 10TH generazione, con memoria RAM da 16GB ed un SSD 512GB.

3.3.1 Risultato Capacity Test

Come richiesto dalla traccia, è stato effettuato un Capacity Test con risorse random (di tutte le tipologie e dunque small, medium e big). Si riportano i grafici di Throughput, Response Time e Potenza ricavati attraverso l'utilizzo di uno script Matlab, che legge i dati prodotti da Python e li plotta nel seguente modo:

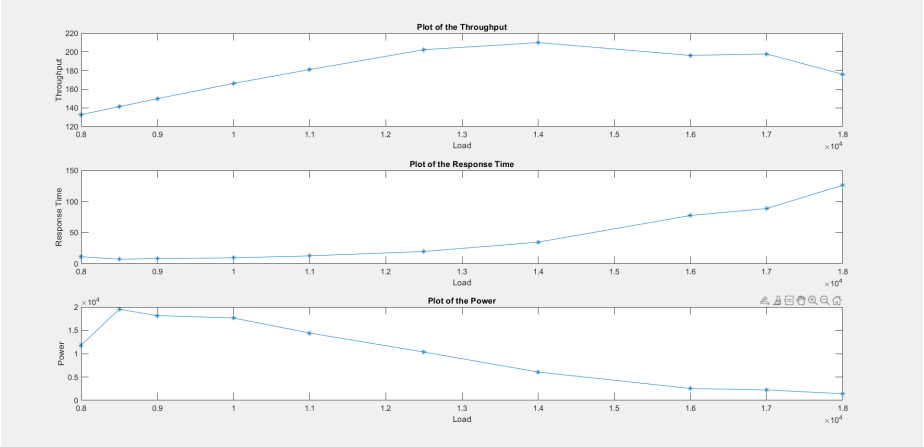


Figura 3.1: **Throughput, Response Time e Potenza.**

Per tale configurazione i punti di lavoro scelti sono i seguenti:
- **Knee Capacity**: 8500 req/s
- **Usable Capacity**: 14000 req/s.

3.4 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 i7-10510U.

Su tale server sono posizionate 43 risorse di dimensioni differenti, di cui la più piccola è 3kB e la più grande è di 5Mb, le quali possono essere richieste dal Client tramite delle richieste HTTP.
Dal sistema appena definito è necessario collezionare due insiemi di features a due livelli di dettaglio differente: le features di alto livello (HL), le quali sono collezionate lato Client tramite il software Apache JMeter; le features di basso livello (LL), raccolte lato Server direttamente dal terminale tramite il comando vmstat.
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.

Di seguito è riportato, in figura 3.2, lo schema delle fasi appena descritte.

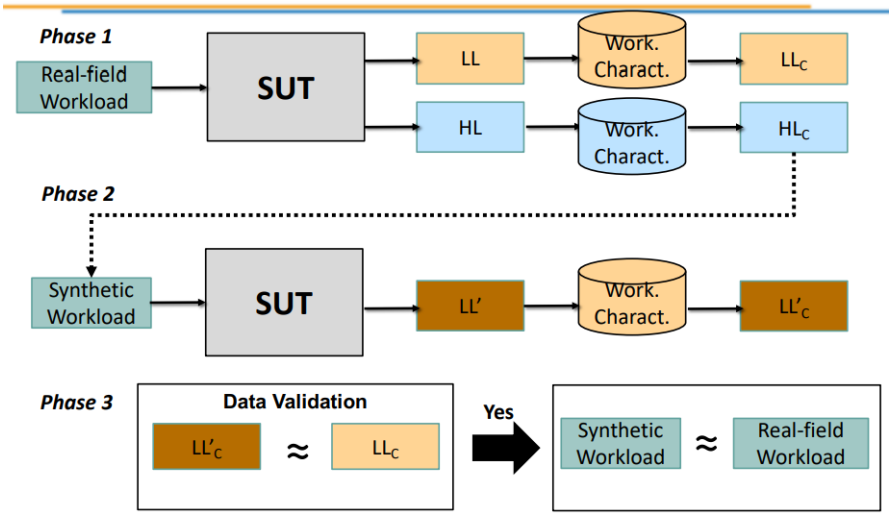


Figura 3.2: **Schema Workload Characterization.**

3.4.1 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, è stato impostato un solo Thread Group, contenente richieste a tutte le risorse del server, effettuate casualmente tramite l'utilizzo di un Random Controller.
Il Thread Group è configurato nel seguente modo:

- 50 utenti
- Ramp-Up period di 50 (si attiva dunque un utente al secondo)
- Durata del test di 300s
- il carico imposto CTT=500.

Per filtrare i dati prodotti da JMeter abbiamo effettuato PCA e Clustering con l'utilizzo di JMP.

	timeStamp	elapsed	label	threadName	dataType	success	failureMessage	bytes	sentBytes	grpThreads	allThreads	URL	Latency	IdleTime	Connect
1	1,638986e+12	3	HTTP Request 27	ThreadGroup1 1-1	text	true		276794	128	1	1	http://192.168.56.104/270kb.txt	1	0	1
2	1,638986e+12	4	HTTP Request 39	ThreadGroup1 1-1	bin	true		399650	128	1	1	http://192.168.56.104/390kb.jpg	1	0	0
3	1,638986e+12	5	HTTP Request 18	ThreadGroup1 1-1	bin	true		184610	128	1	1	http://192.168.56.104/180kb.jpg	2	0	0
4	1,638986e+12	3	HTTP Request 24	ThreadGroup1 1-1	text	true		246073	128	1	1	http://192.168.56.104/240kb.txt	1	0	0
5	1,638986e+12	11	HTTP Request 33	ThreadGroup1 1-1	bin	true		338209	128	1	1	http://192.168.56.104/330kb.png	3	0	0
6	1,638986e+12	8	HTTP Request 43	ThreadGroup1 1-1	text	true		819513	128	1	1	http://192.168.56.104/800kb.txt	2	0	0
7	1,638986e+12	8	HTTP Request 24	ThreadGroup1 1-1	text	true		246073	128	1	1	http://192.168.56.104/240kb.txt	3	0	0
8	1,638986e+12	4	HTTP Request 43	ThreadGroup1 1-1	text	true		819513	128	1	1	http://192.168.56.104/800kb.txt	1	0	0
9	1,638986e+12	1	HTTP Request 19	ThreadGroup1 1-1	text	true		194873	128	1	1	http://192.168.56.104/190kb.txt	1	0	0
10	1,638986e+12	2	HTTP Request 16	ThreadGroup1 1-2	bin	true		168843	128	2	2	http://192.168.56.104/160kb.jpg	1	0	1
11	1,638986e+12	2	HTTP Request 31	ThreadGroup1 1-1	bin	true		317729	128	2	2	http://192.168.56.104/310kb.png	1	0	0
12	1,638986e+12	6	HTTP Request 35	ThreadGroup1 1-2	bin	true		358689	128	2	2	http://192.168.56.104/350kb.png	2	0	0
13	1,638986e+12	3	HTTP Request 10	ThreadGroup1 1-1	text	true		92472	127	2	2	http://192.168.56.104/90kb.txt	2	0	0
14	1,638986e+12	4	HTTP Request 42	ThreadGroup1 1-2	bin	true		678714	128	2	2	http://192.168.56.104/680kb.jpg	1	0	0
15	1,638986e+12	2	HTTP Request 4	ThreadGroup1 1-1	bin	true		4442	126	2	2	http://192.168.56.104/5kb.png	2	0	0
16	1,638986e+12	6	HTTP Request 12	ThreadGroup1 1-2	text	true		123193	128	2	2	http://192.168.56.104/120kb.txt	3	0	0
17	1,638986e+12	4	HTTP Request 14	ThreadGroup1 1-1	bin	true		143650	128	2	2	http://192.168.56.104/140kb.jpg	2	0	0
18	1,638986e+12	5	HTTP Request 43	ThreadGroup1 1-2	text	true		819513	128	2	2	http://192.168.56.104/800kb.txt	1	0	0
19	1,638986e+12	2	HTTP Request 15	ThreadGroup1 1-3	bin	true		153891	128	3	3	http://192.168.56.104/150kb.jpg	1	0	1
20	1,638986e+12	6	HTTP Request 42	ThreadGroup1 1-1	bin	true		678714	128	3	3	http://192.168.56.104/680kb.jpg	2	0	0
21	1,638986e+12	3	HTTP Request 33	ThreadGroup1 1-2	bin	true		338209	128	3	3	http://192.168.56.104/330kb.png	1	0	0
22	1,638986e+12	4	HTTP Request 29	ThreadGroup1 1-3	bin	true		297249	128	3	3	http://192.168.56.104/290kb.png	1	0	0
23	1,638986e+12	1	HTTP Request 12	ThreadGroup1 1-1	text	true		123193	128	3	3	http://192.168.56.104/120kb.txt	1	0	0
24	1,638986e+12	5	HTTP Request 24	ThreadGroup1 1-2	text	true		246073	128	3	3	http://192.168.56.104/240kb.txt	2	0	0
25	1,638986e+12	2	HTTP Request 23	ThreadGroup1 1-3	text	true		235833	128	3	3	http://192.168.56.104/230kb.txt	1	0	0
26	1,638986e+12	6	HTTP Request 23	ThreadGroup1 1-1	text	true		235833	128	3	3	http://192.168.56.104/230kb.txt	2	0	0
27	1,638986e+12	29	HTTP Request 5	ThreadGroup1 1-2	text	true		5243195	126	3	3	http://192.168.56.104/5MB.txt	2	0	0
28	1,638986e+12	2	HTTP Request 2	ThreadGroup1 1-4	bin	true		75042	126	4	4	http://192.168.56.104/3kb.jpg	1	0	1
29	1,638986e+12	3	HTTP Request 19	ThreadGroup1 1-3	text	true		194873	128	4	4	http://192.168.56.104/190kb.txt	2	0	0
30	1,638986e+12	3	HTTP Request 19	ThreadGroup1 1-1	text	true		194873	128	4	4	http://192.168.56.104/190kb.txt	1	0	0
31	1,638986e+12	10	HTTP Request 43	ThreadGroup1 1-2	text	true		819513	128	4	4	http://192.168.56.104/800kb.txt	3	0	0
32	1,638986e+12	8	HTTP Request 33	ThreadGroup1 1-4	bin	true		338209	128	4	4	http://192.168.56.104/330kb.png	4	0	0
33	1,638986e+12	9	HTTP Request 1	ThreadGroup1 1-3	text	true		1048891	126	4	4	http://192.168.56.104/1MB.txt	2	0	0
34	1,638986e+12	29	HTTP Request 5	ThreadGroup1 1-1	text	true		5243195	126	4	4	http://192.168.56.104/5MB.txt	2	0	0
35	1,638986e+12	5	HTTP Request 35	ThreadGroup1 1-2	bin	true		358689	128	4	4	http://192.168.56.104/350kb.png	2	0	0
36	1,638986e+12	1	HTTP Request 23	ThreadGroup1 1-4	text	true		235833	128	4	4	http://192.168.56.104/230kb.txt	0	0	0
37	1,638986e+12	5	HTTP Request 10	ThreadGroup1 1-5	text	true		92473	127	5	5	http://192.168.56.104/90kb.txt	4	0	2
38	1,638986e+12	4	HTTP Request 22	ThreadGroup1 1-3	text	true		225593	128	5	5	http://192.168.56.104/220kb.txt	2	0	0

Figura 3.3: **Workload Prodotto.**

In figura 3.3 abbiamo riportato un estratto del workload caratterizzato da 2549 righe.

Le colonne **Timestamp, Success, Failur message, Idle Time** sono state eliminate in quanto costanti, per cui il loro contributo alla varianza spiegata risulta essere nullo.
Inizialmente avevamo scelto di considerare 4 componenti principali e 20 cluster, perdendo il 15,5% della varianza totale spiegata, ma andando ad osservare la composizione dei cluster abbiamo notato la presenza di un outlier alla riga 616. Abbiamo dunque scelto di eliminare tale riga, ma ripetendo i passaggi precedenti è emerso un ulteriore outlier alla riga 1, la quale è stata anch'essa eliminata. Dunque in conclusione abbiamo scelto di prendere 20 cluster, 4 componenti principali con varianza spiegata pari ad 83%.

In figura 3.4 è possibile vedere il risultato di tale operazione.

Principale1	Principale2	Principale3	Principale4	Cluster
-3,999637417	-2,877983575	1,9995791924	-1,210445185	4
-4,20909815	-2,677617482	2,0257771015	0,3203074557	1
-3,901759712	-3,340163163	1,4934923613	-2,474174681	4
-3,755787242	-2,557988659	2,8447413867	0,070415204	1
-4,106564451	-2,394247868	2,5188466591	0,038780998	1
-4,113585091	-2,833024824	1,9838591575	0,4310367838	1
-3,913108272	-2,061810098	2,509696416	-0,209808312	1
-3,781258534	-2,785981321	2,2781422879	0,3694864076	1
-4,281611745	-2,338051574	2,0729098939	0,1404764818	1
-4,230731191	-2,387141246	2,0960006096	0,1796735293	1
-4,456288075	-2,066183205	2,1329317942	-0,054778452	1
-3,948435939	-2,459864625	2,4244416228	0,1439010127	1
-4,367689703	-2,205119289	2,1010853122	0,0448687423	1
-3,890011821	-2,526923579	2,436544899	0,1951880896	1
-0,897956017	-8,30947897	-0,218532549	1,090621258	17
-3,499565564	-3,121786051	1,4134226588	-2,578881415	4
-4,01084681	-2,216475369	2,3596596217	0,0226105682	1
-4,244770136	-2,18686594	2,0579632201	0,0935595678	1
-3,196967219	-3,122801217	2,4970303266	0,5847117742	1
-3,20310956	-2,724780417	2,9539902724	0,2137086947	1
-3,074637664	-4,327755602	1,0465904628	-1,656918718	4
-0,825917004	-8,24213595	-0,284745357	1,100115036	17
-3,848743086	-2,458394935	2,3179501206	0,1977154533	1
-4,590512655	-2,035118124	1,7247353065	0,0699944332	1
-2,244390196	-2,690514916	3,1798216	-1,65808392	4
-3,870325595	-2,240150346	2,2909001706	0,0995114624	1
-4,240061901	-2,031247694	1,9926199688	0,0374916121	1
-3,390780709	-3,240792245	1,6049911358	-2,46132526	4
-4,175251292	-2,113519705	1,9954211184	0,0990142842	1
-3,435990597	-2,530820209	2,5922901861	0,2227143463	1
-3,419337894	-4,011657007	0,6533588646	-1,7424272	4
-3,980504638	-3,035475957	0,98471674	-2,430061434	4
-4,221097156	-2,076423009	1,9649972168	0,0678863082	1
-1,093791238	-2,70480302	4,2997123917	-0,224921866	2
-3,891129426	-3,009428945	0,8932532081	-2,392783021	4
-3,762170736	-2,186726456	2,2692833447	0,1245389675	1
-3,22587388	-3,031554462	2,0713491058	0,7299654165	1

Figura 3.4: **PCA e Clustering lato Client.**

Per determinare il workload sintetico lato client è possibile procedere in due modi differenti:
1) Selezionando in modo random una riga per ogni cluster;
2) Selezionando per ogni cluster la richiesta rappresentativa.

Abbiamo scelto di utilizzare il secondo metodo.
Nella tabella in figura 3.5 riportiamo la composizione di ogni cluster e la sua richiesta rappresentativa (nel caso in cui più richieste fossero rappresentative per lo stesso cluster è stata selezionata la richiesta che rispetto al suo totale è contenuta maggiormente in quel cluster).

A valle di questo procedimento le richieste rappresentative sono :$1, 2, 3, 3, 3, 4, 5, 10, 9, 10, 12, 17, 19, 21, 22, 26, 35, 43, 41, 43$.

CLUSTER	RIGHE TOTALI	RICHIESTA SCELTA
1	76	43
2	26	35
3	147	41
4	54	9
5	182	10
6	39	1
7	64	2
8	60	4
9	69	(10,9) 7
10	87	17
11	199	26
12	804	22
13	187	19
14	34	(43) 36
15	382	12
16	5	21
17	22	3
18	36	(3) 1
19	42	5
20	32	3

Figura 3.5: **Tabella descrittiva dei Cluster.**

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 305 secondi (ovvero abbiamo un riga al secondo per un totale di 305 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	58496	387808	159536	2440592	0	0	0	80	308	535	1	0	99	0	0
0	0	58496	387808	159536	2440592	0	0	0	0	158	268	0	0	100	0	0
0	0	58496	387648	159536	2440592	0	0	0	0	386	282	1	0	99	0	0
0	0	58496	387776	159536	2440592	0	0	0	0	702	427	1	1	98	0	0
0	0	58496	389056	159560	2440496	0	0	0	1192	862	523	1	1	98	1	0
0	0	58496	389056	159560	2440496	0	0	0	0	1172	386	1	2	97	0	0
0	0	58496	389192	159560	2440496	0	0	0	0	1176	416	0	2	98	0	0
0	0	58496	389372	159560	2440496	0	0	0	0	704	409	2	2	96	1	0
0	0	58496	389268	159560	2440500	0	0	0	0	1072	342	0	2	98	0	0
1	0	58496	389204	159568	2440500	0	0	0	20	708	553	2	2	96	0	0
0	0	58496	389268	159568	2440500	0	0	0	0	503	397	1	0	99	0	0
1	0	58496	389172	159568	2440504	0	0	0	0	557	472	2	2	97	0	0
0	0	58496	389332	159568	2440504	0	0	0	0	515	366	1	2	97	0	0
0	0	58496	389332	159568	2440504	0	0	0	0	501	470	1	1	98	0	0
0	0	58496	389332	159576	2440508	0	0	0	12	820	485	1	2	97	0	0
0	0	58496	389316	159576	2440508	0	0	0	148	1020	408	1	2	97	0	0
1	0	58496	389276	159576	2440508	0	0	0	0	581	412	2	2	96	0	0
0	0	58496	389244	159576	2440508	0	0	0	0	515	424	1	2	97	0	0
0	0	58496	389276	159576	2440512	0	0	0	0	548	409	1	1	98	0	0
0	0	58496	389292	159576	2440512	0	0	0	0	797	411	1	2	97	0	0
0	0	58496	389276	159584	2440512	0	0	0	236	1102	447	1	1	98	0	0
0	0	58496	389308	159584	2440516	0	0	0	0	834	377	1	2	97	0	0
0	0	58496	389412	159584	2440516	0	0	0	0	1118	442	2	1	97	0	0
0	0	58496	389244	159584	2440516	0	0	0	0	802	435	1	2	98	0	0
0	0	58496	389340	159584	2440516	0	0	0	0	443	372	1	1	98	0	0
0	0	58496	389276	159592	2440520	0	0	0	12	774	468	1	1	98	1	0
0	0	58496	389340	159592	2440520	0	0	0	0	853	417	2	3	96	0	0
0	0	58496	389292	159592	2440520	0	0	0	0	665	378	2	1	97	0	0
0	0	58496	389308	159592	2440524	0	0	0	0	1532	447	1	3	96	0	0

Figura 3.6: **Workload lato Server.**

In prima battuta, sono state eliminate le feature costanti b, swpd, si, so, st, in quanto non contribuiscono alla varianza del dataset. Dopo aver effettuato un'analisi dei dati, sono stati riscontrati ed eliminati 5 outlier(alla rispettive righe 1, 296, 297, 298, 127).
A valle di queste modifiche, sono state prese 7 componenti principali e 40 cluster.

3.4.2 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 (4 componenti principali, 20 cluster).

3.4.3 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 statisiticamente, 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 KS-test, dal quale è emerso che la distribuzione non è normale. Dunque si è proceduto ad effettuare il test non parametrico noto come Somma dei ranghi, o test delle Mediane. 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-valuerif = 0.05, in modo da poter affermare che non ci siano differenze statistiche tra i dataset ed accettare di conseguenza l'ipotesi nulla H0 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.

Riportiamo di seguito, in figura 3.7, i risultati ottenuti attraverso uno script Matlab, dove è possibile osservare sia che l'ipotesi nulla è stata accettata sia i valori di p-value prodotti.

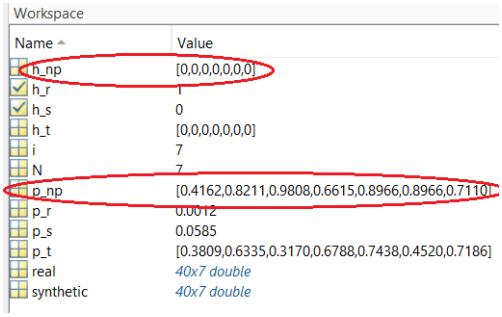


Figura 3.7: **Risultati Workload Characterization.**

3.5 Design of Experiments

L'ultimo studio proposto è quello del Design of Experiment, il quale si propone di verificare l'influenza dei fattori controllabili sul Response Time del sistema. Nel contesto di tale esercizio, sono stati scelti 2 fattori differenti:

- **Load** (carico) imposto al sistema server da parte del Client, espresso in termini di richieste al secondo. Sono stati individuati 2 differenti livelli di carico nel seguente modo:
– Low intensity: 25% della Usable Capacity (3500 req/sec)
– High intesity: 75% della Usable Capacity (10500 req/s)

- **Tipologia di risorsa richiesta**:
- Livello 1: small = 13kb.jpg
- Livello 2: medium-small = 400kb.jpg
- Livello 3: medium-high= 1MB.txt
- Livello 4: high = 3MB.txt

Sulla base di tali fattori e livelli con 5 ripetizioni per ogni combinazione della durata di 1 minuto abbiamo realizzato 40 esperimenti.
In figura 3.8 si riportano i risultati degli esperimenti condotti.

	Intensita	Pagina	Y
1	10500	1	1,20552496677425
2	10500	1	1,36772235918831
3	10500	1	1,09767662399241
4	10500	1	1,03813318155947
5	10500	1	0,986724824578039
6	3500	1	1,14890016920473
7	3500	1	1,41650239369191
8	3500	1	1,6034919740918
9	3500	1	1,88256829062236
10	3500	1	1,6865671641791
11	10500	2	0,816233643087426
12	10500	2	0,901090564248459
13	10500	2	0,784901365705614
14	10500	2	0,684714583728427
15	10500	2	0,784981963166888
16	3500	2	1,16248943959448
17	3500	2	1,22810475922275
18	3500	2	0,95071810757533
19	3500	2	1,01408054069276
20	3500	2	1,00732188116023
21	10500	3	336,68679031037
22	10500	3	296,050239234449
23	10500	3	336,167227833894
24	10500	3	319,541910331384
25	10500	3	323,076238881829
26	3500	3	7,24021402421853
27	3500	3	9,67915492957746
28	3500	3	127,494782608695
29	3500	3	63,4432989690721
30	3500	3	100,825925925925
31	10500	4	736,854523227383
32	10500	4	785,597265108072
33	10500	4	858,023200757575
34	10500	4	946,807153965785
35	10500	4	1061,03267211201
36	3500	4	544,147657512116
37	3500	4	589,904509283819
38	3500	4	600,249409382382
39	3500	4	601,328246863343
40	3500	4	662,170577281191

Figura 3.8: **Esperimenti condotti.**

Nella tabella la colonna "Intensità" fa riferimento al carico che è stato imposto; la colonna "Pagina" fa riferimento al tipo di pagina (caratterizzato da una determinata dimensione); infine la "Y" rappresenta i nostri tempi di risposta.

Per lo sviluppo di questo elaborato ci siamo interessate nel determinare l'importanza e la significatività dei fattori che abbiamo analizzato:

1) Importanza: un fattore si definisce importante se esprime una buona percentuale di variazione rispetto alla variazione totale del sistema. L'importanza non è un concetto statistico, siamo noi analiste dei dati che decidiamo ciò che per noi è importante;

2) Significatività: rappresenta il contributo che spiega un fattore rispetto a quello spiegato dall'errore. Quando definiamo fattore significativo siamo in grado di affermare che ripetendo l'esperimento ci possiamo ritrovare nelle stesse condizioni di prima. La significatività non è di conseguenza un concetto soggettivo.

3.5.1 Analisi dell'importanza

Per calcolare l'importanza dei fattori, in base alla definizione data precedentemente, andiamo a valutare la percentuale di variazione spiegata dal Intensità (SSA), dalla Pagina (SSB), dall'interazione di questi ultimi (SSAB) e dell'errore (SSE).

Analisi della varianza

Origine	DF	Somma dei quadrati	Media quadratica	Rapporto F
Modello	7	4011510,4	573073	211,2382
Errore	32	86813,5	2713	Prob > F
C. totale	39	4098324,0		<,0001*

Stime dei parametri

Test degli effetti

Origine	Nparm	DF	Somma dei quadrati	Rapporto F	Prob > F
Intensita	1	1	180891,8	66,6778	<,0001*
Pagina	3	3	3648417,8	448,2764	<,0001*
Intensita*Pagina	3	3	182200,8	22,3868	<,0001*

Figura 3.9: **Analisi della varianza.**

$$\frac{SSA}{SST} = \frac{18.08911,8}{4098324,0} = 0.0441$$
$$\frac{SSB}{SST} = \frac{3648417,8}{4098324,0} = 0.8902$$
$$\frac{SSAB}{SST} = \frac{182200,8}{4098324,0} = 0.0444$$
$$\frac{SSE}{SST} = \frac{86813,5}{4098324,0} = 0.0211$$

Il fattore B, cioè il fattore Pagina, che fa riferimento alla dimensione, risulta essere il fattore più importante. L'errore è il meno importante.

3.5.2 Analisi della significatività

Prima di procedere con lo studio della significatività, è necessario analizzare due assunzioni fondamentali:
- normalità dei residui: gli errori devono essere normalmente distribuiti;
- omoschedasticità: la deviazione standard degli errori deve essere costante.

Tali ipotesi vanno verificate necessariamente per definire la tipologia di test da poter effettuare, come risulta dalla seguente tabella:

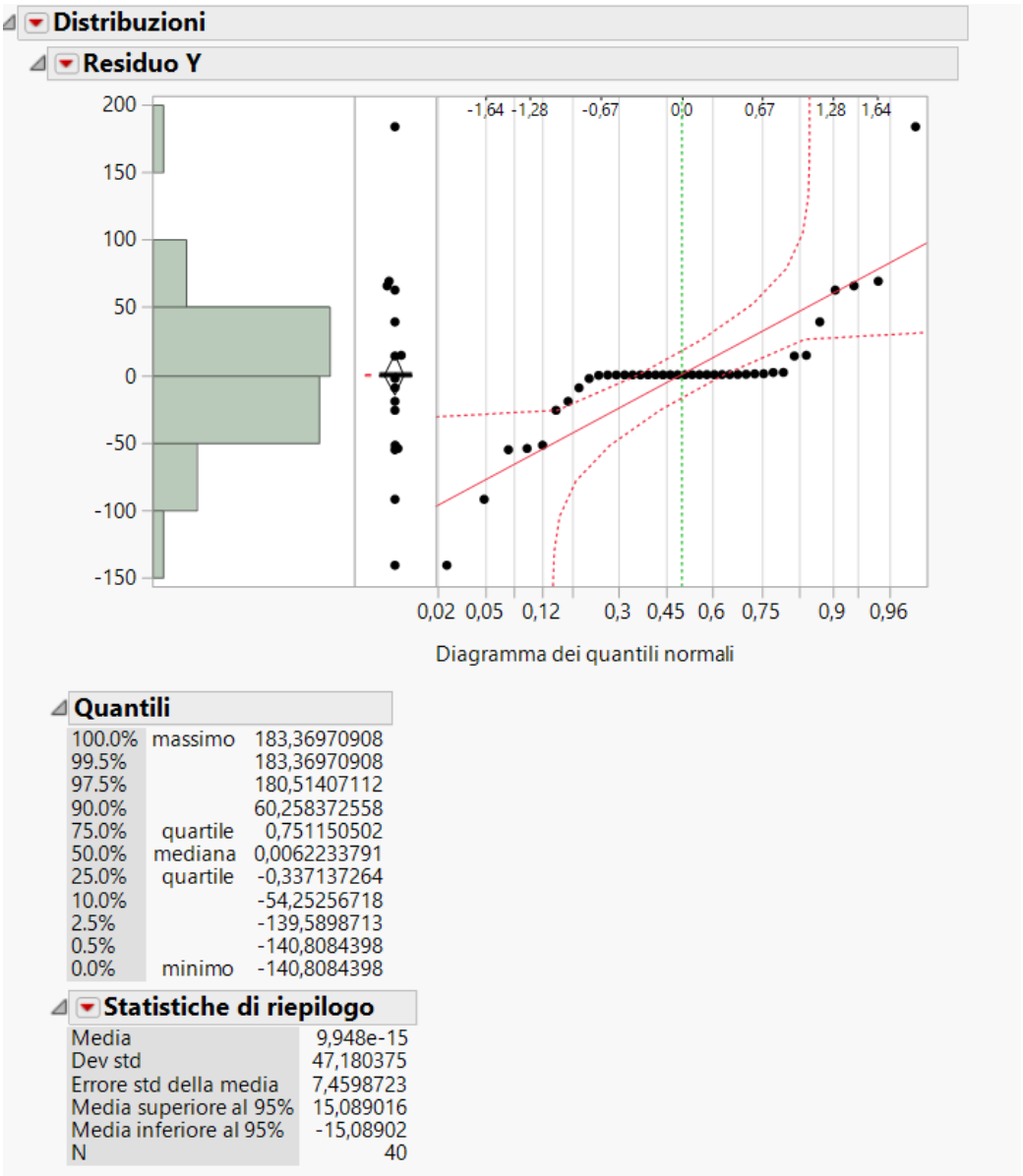
Normalità	Omoschedasticità	Test ANOVA
✔	✔	*F-Test*
✘	✔	*Kruskal-Wallis test*
✔	✘	*Welch test*
✘	✘	*Kruskal-Wallis / Friedman test*

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

	Intensita	Pagina	Y	Residuo Y
1	10500	1	1,20552496677425	0,0663685756
2	10500	1	1,36772235918831	0,228565968
3	10500	1	1,09767662399241	-0,041479767
4	10500	1	1,03813318155947	-0,10102321
5	10500	1	0,986724824578039	-0,152431567
6	3500	1	1,14890016920473	-0,398705829
7	3500	1	1,41650239369191	-0,131103605
8	3500	1	1,6034919740918	0,0558859757
9	3500	1	1,88256829062236	0,3349622923
10	3500	1	1,6865671641791	0,1389611658
11	10500	2	0,816233643087426	0,0218492191
12	10500	2	0,901090564248459	0,1067061403
13	10500	2	0,784901365705614	-0,009483058
14	10500	2	0,684714583728427	-0,10966984
15	10500	2	0,784981963166888	-0,009402461
16	3500	2	1,16248943959448	0,0899464939
17	3500	2	1,22810475922275	0,1555618136
18	3500	2	0,95071810757533	-0,121824838
19	3500	2	1,01408054069276	-0,058462405
20	3500	2	1,00732188116023	-0,065221064
21	10500	3	336,68679031037	14,382308992
22	10500	3	296,050239234449	-26,25424208
23	10500	3	336,167227833894	13,862746516
24	10500	3	319,541910331384	-2,762570987
25	10500	3	323,076238881829	0,7717575634
26	3500	3	7,24021402421853	-54,49646127
27	3500	3	9,67915492957746	-52,05752036
28	3500	3	127,494782608695	65,758107317
29	3500	3	63,4432989690721	1,7066236776
30	3500	3	100,825925925925	39,089250634
31	10500	4	736,854523227383	-140,8084398
32	10500	4	785,597265108072	-92,06569793
33	10500	4	858,023200757575	-19,63976228
34	10500	4	946,807153965785	69,144190932
35	10500	4	1061,03267211201	183,36970908
36	3500	4	544,147657512116	-55,41242255
37	3500	4	589,904509283819	-9,655570781
38	3500	4	600,249409382382	0,6893293178
39	3500	4	601,328246863343	1,7681667988
40	3500	4	662,170577281191	62,610497217

Figura 3.10: **Calcolo dei residui.**

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



Il test visivo permette di rigettare l'ipotesi di normalità, in quanto tutti i residui non 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 conferma il risultato visivo, come si può notare dalla seguente immagine in figura 3.11, permettendoci di fatto di non confermare la normalità dei residui.

Stima normale

Stime dei parametri

Tipo	Parametro	Stima	Inferiore al 95%	Superiore al 95%
Percorso	μ	9,948e-15	-15,08902	15,089016
Dispersione	σ	47,180375	38,648324	60,581276

-2log(Likelihood) = 420,833324041535

Test della bontà di adattamento

Test W di Shapiro-Wilk

W	Prob<W
0,745392	<,0001*

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

Figura 3.11: **Test di normalità.**

Di conseguenza, per calcolare la significatività dei fattori sarà utilizzato un test non parametrico.

Per quanto riguarda l'omoschedasticità, essa va verificata per entrambi i fattori; JMP mette a disposizione molteplici test, tra i quali è stato scelto quello di Levene.

Intensita

Livello	Conteggio	Dev std	Diff assoluta media rispetto alla media	Diff assoluta media rispetto alla mediana
3500	20	259,9673	216,7904	164,6692
10500	20	372,2870	299,9510	299,5085

Test	Rapporto F	Num DF	Den DF	p-value
O'Brien[.5]	3,0194	1	38	0,0904
Brown-Forsythe	2,6959	1	38	0,1089
Levene	2,2308	1	38	0,1435
Bartlett	2,3378	1	.	0,1263
Test F bilaterale	2,0508	19	19	0,1263

Figura 3.12: **Test omoschedasticità, fattore Intensità.**

Pagina

Livello	Conteggio	Dev std	Diff assoluta media rispetto alla media	Diff assoluta media rispetto alla mediana
1	10	0,3018	0,2480	0,2480
2	10	0,1739	0,1391	0,1391
3	10	142,3495	130,2839	130,2839
4	10	172,4125	139,4028	139,0514

Test	Rapporto F	Num DF	Den DF	Prob > F
O'Brien[.5]	6,2234	3	36	0,0016*
Brown-Forsythe	20,5148	3	36	<,0001*
Levene	25,4053	3	36	<,0001*
Bartlett	67,1250	3	.	<,0001*

Figura 3.13: **Test omoschedasticità, fattore Pagina.**

Nel caso del fattore Intensità, l'omoschedasticità è verificata, mentre nel caso del fattore Pagina non lo è.
Considerando anche il fatto che l'ipotesi di normalità non è verificata, bisognerà applicare i sequenti test per la significatività:

- Intensity, ANOVA non-parametrico omoschedastico (test di Kruskal-Wallis)
- PageType, ANOVA non-parametrico eteroschedastico (test di Kruskal-Wallis)

Test di Wilcoxon/Kruskal-Wallis (somme dei ranghi)

Livello	Conteggio	Somma degli score	Score atteso	Media degli score	(Media-Media0)/Std0
3500	20	417,000	410,000	20,8500	0,176
10500	20	403,000	410,000	20,1500	-0,176

Test a due campioni, approssimazione normale

S	Z	Prob>|Z|
403	-0,17583	0,8604

Test a una variabile, approssimazione chi-quadrato

Chi-quadrato	DF	Prob>ChiQu
0,0359	1	0,8498

Figura 3.14: **Test di Kruskal-Wallis per fattore Intensità.**

Test di Wilcoxon/Kruskal-Wallis (somme dei ranghi)

Livello	Conteggio	Somma degli score	Score atteso	Media degli score	(Media-Media0)/Std0
1	10	144,000	205,000	14,4000	-1,890
2	10	66,000	205,000	6,6000	-4,326
3	10	255,000	205,000	25,5000	1,546
4	10	355,000	205,000	35,5000	4,670

Test a una variabile, approssimazione chi-quadrato

Chi-quadrato	DF	Prob>ChiQu
35,1527	3	<,0001*

Figura 3.15: **Test di Kruskal-Wallis per fattore Pagina.**

Nel test di Kruskal-Wallis i p-value bassi indicano che il corrispondente fattore è significativo, quindi risulta essere che il fattore Intensità non risulta essere significativo, mentre il fattore Pagina è significativo.

Di seguito una tabella che riassume i risultati ottenuti:

	Importanza	Significatività
Intensità	4.41%	NO
Pagina	89.02%	SI
Interazione	4,4%	NON APPLICABILE

Figura 3.16: **Tabella risultati DOE.**

Capitolo 4

RBD

4.1 Esercizio 1

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

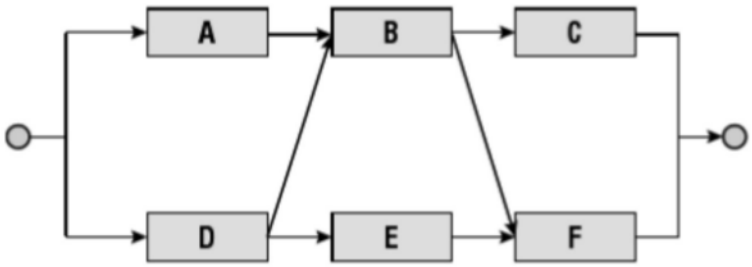


Figura 4.1: **Traccia esercizio 1.**

Il Reliability Block Diagram in figura non è descrivibile come una combinazione di serie e parallelo di componenti in modo immediato; per questo motivo, si è ritenuto opportuno utilizzare il teorema della probabilità totale per risolvere l'esercizio. Abbiamo deciso di operare in primo luogo con il nodo E: per cui abbiamo espresso la reliability del sistema, come il prodotto tra la reliability del componente E per la probabilità che il sistema funzioni dato che il componente E funzioni, sommando a questa quantità, il prodotto della unreliability del componente E per la probabilità che il sistema funzioni dato che il componente E sia fallito. Riportiamo la formula che riassume quanto detto:

$$R_{sys} = R_E P(systemworks|Eworks) + (1 - R_E)P(systemworks|Efails)$$

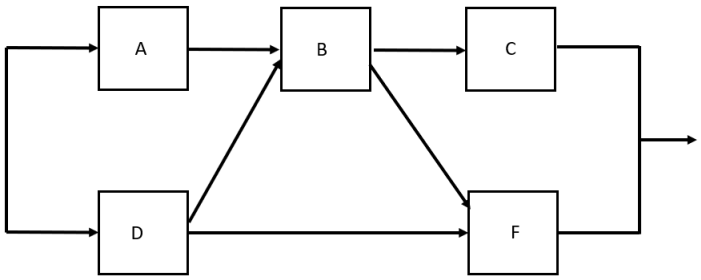


Figura 4.2: **Reliability Block Diagram con E funzionante.**

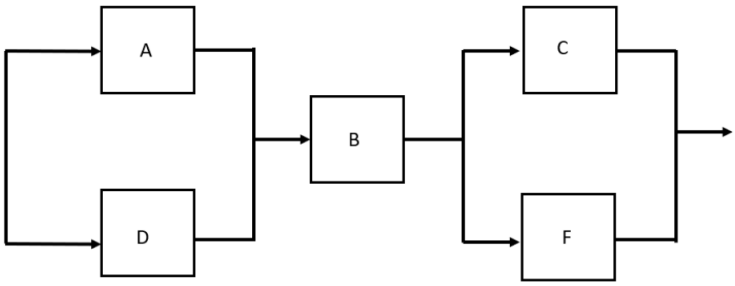


Figura 4.3: **Reliability Block Diagram con E non funzionante.**

Il Reliability Block Diagram risultante, dato il componente E non funzionante (figura 4.3), si può vedere come una combinazione semplice ed immediata di serie e parallelo di componenti. L'espressione della Reliability è dunque la seguente: abbiamo il componente A in parallelo con il componente D, in serie con il componente B, in serie con il parallelo tra il componente C ed il componente F. Riportiamo la formula che riassume quanto detto:

$$R_{sysEfails} = [1 - (1 - R_A)(1 - R_D)]R_B[1 - (1 - R_C)(1 - R_F)]$$

Poichè nella traccia è specificato che tutti i componenti hanno una reliability pari ad R:

$$R_{sysEfails} = R_5 - 4R_4 + 4R_3$$

Per il Reliability Block Diagram risultante, dato il componente E funzionante (figura 4.2), si applica nuovamente il teorema della probabilità totale operando

sul componente B, in quanto la semplificazione serie e parallelo non risulta immediata. Esprimiamo quindi la reliability del sistema come: il prodotto tra la reliability del componente B per la probabilità che il sistema funzioni dato che il componente B funzioni, sommando a questa quantità, il prodotto della unreliability del componente B per la probabilità che il sistema funzioni dato che il componente B sia fallito. Riportiamo la formula che riassume quanto detto:

$$R_{sys} = R_B P(systemworks|Bworks) + (1 - R_B)P(systemworks|Bfails)$$

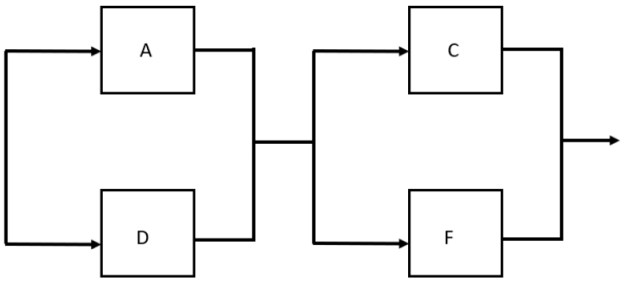


Figura 4.4: **Reliability Block Diagram con EBfunzionante.**

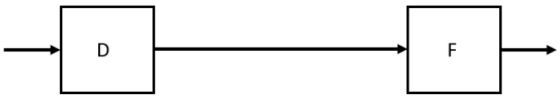


Figura 4.5: **Reliability Block Diagram con B non funzionante.**

Per il Reliability Block Diagram risultante, dato il componente B funzionante (figura 4.4), abbiamo che:

$$R_{sysEworksBworks} = [1-(1-R_A)(1-R_D)][1-(1-R_C)(1-R_F)] = R^4-4R^3+4R^2$$

Per il Reliability Block Diagram risultante, dato il componente B non funzionante (figura 4.5), abbiamo che:

$$R_{sysEworksBfails} = R_D R_F = R^2$$

Avendo adesso a disposizione l'espressione della Reliability del sistema sia quando E funziona che quando fallisce, è possibile calcolare la Reliability del sistema iniziale.

$$R_{sys} = R_E R_{sysEworks} + (1 - R_E)R_{sysEfails}$$

$$R_{sys} = R(R^5 - 4R^4 + 3R^3 + R^2) + (1 - R)(R^5 - 4R^4 + 4R^3) = R^5 - 5R^4 + 5R^3$$

Considerando che, come richiede la traccia, il tempo di fallimento è una variabile aleatoria che segue una distribuzione esponenziale con tasso di fallimento λ, si ottiene:

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

Da cui si può facilmente ricavare l'MTTF:

$$MTTF_{sys} = \frac{1}{5\lambda} - \frac{5}{4\lambda} + \frac{5}{3\lambda} = \frac{37}{60\lambda}$$

4.2 Esercizio 2

We want to compare two different schemes of increasing reliability of a system using redundancy. Suppose that the system needs *s* identical components in series for proper operation. Further suppose that we are given (*m x s*) components. Given that the reliability of an individual component is r, derive the expressions for the reliabilities of two configurations. For *m* = 3 and *s* =4, compare the two expressions as function of a mission time t.
Let MTTF of the component be 100 hours.
Out of the two schemes shown in the figure below, which one will provide a higher reliability? Modify the scheme that has lower reliability in order to reach the same reliability of the other.

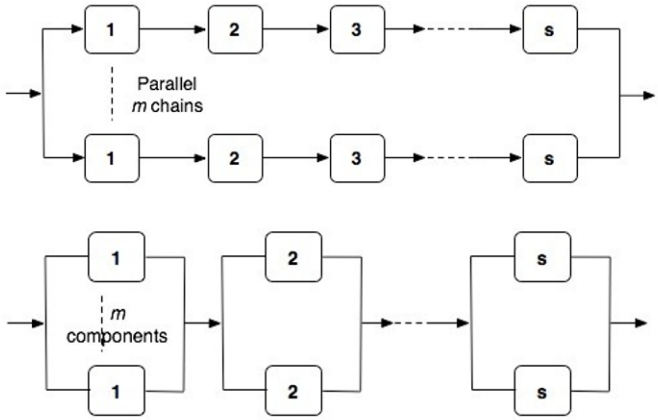


Figura 4.6: **Traccia esercizio 2.**

Dati m = 3 e s = 4, la prima configurazione, a cui si farà riferimento da adesso in poi con sys1, è costituita da tre catene formate ciascuna da quattro componenti in serie. Dunque, l'espressione della Reliability è la seguente:

$$R_{sys1} = [1 - (1 - R^4)^3]$$

Nel caso della seconda configurazione, a cui si farà riferimento da adesso in poi con sys2, è costituita da quattro blocchi in serie ciascuno formato dal parallelo di tre componenti. Dunque, l'espressione della Reliability è la seguente:

$$R_{sys2} = [1 - (1 - R)^3]^4$$

La comparazione della Reliability delle due configurazioni per m = 3 e s = 4, in funzionedel tempo, supponendo un MTTF del componente pari a 100 ore:

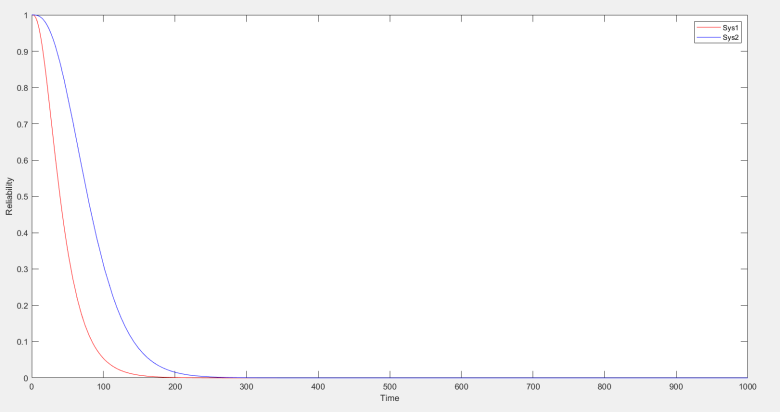


Figura 4.7: **Comparazione della Reliability delle due configurazioni per m = 3 e s = 4.**

Vediamo come per m = 3 e s = 4, utilizzando lo stesso componente con una data Reliability r in entrambe le configurazioni, il sistema Sys2 ha una maggiore Reliability per qualsiasi valore del Mission Time rispetto al sistema Sys1.

A valle di un'analisi visiva per differenti valori di m, è stato trovato il valore di m per il quale le due espressioni di Reliability sono più simili tra loro. Tale valore è 10. Dunque, partendo da m = 3 e s = 4 per entrambe le configurazioni, la modifica da attuare sul Sys1 per raggiungere una Reliability quanto più simile possibile a quella del Sys2 è di avere in totale 10 catene da quattro componenti.. Quest'analisi visiva, l'abbiamo confermata anche numericamente:

$$1 - (1 - R^4)^n = [1 - (1 - R)^3]^4$$

$$n = \frac{\ln\left(1 - [1 - (1 - R)^3]^4\right)}{\ln\left(1 - R^4\right)}$$

Quindi per calcolare il numero di paralleli necessari ad uguagliare la reliability dei due sistemi, serve determinare il valore di R, e di conseguenza il valore del mission time a cui facciamo riferimento, in quanto senza la definizione di mission time non esiste la definizione di reliability.

Supponendo di avere tempi di fallimenti esponenziali, e che il mission time sia pari a 48h, otteniamo che $R(t) = e^{(-\frac{48}{100})} = 0.62$.
Inserendo tale valore nella formula ricavata precedentemente, otteniamo che $n = 10.0$; di conseguenza è anche matematicamente giustificato il valore scelto dall'analisi visiva.
Successivamente, abbiamo quindi modificato la formula della reliability del sistema 1 ed abbiamo nuovamente plottato i due andamenti. Riportiamo il risultato in figura 4.8:

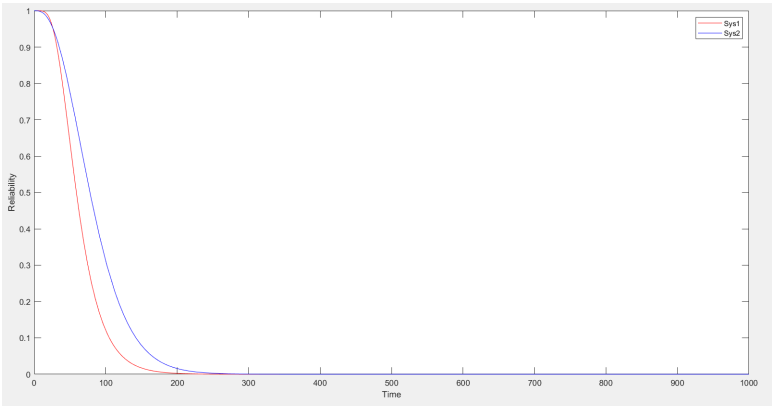


Figura 4.8: **Comparazione della Reliability delle due configurazioni per m = 7 e s = 4.**

Possiamo affermare che per valori iniziali del Mission Time è preferibile il Sys1, mentre per Mission Time più elevati è preferibile il Sys2.

4.3 Esercizio 3

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

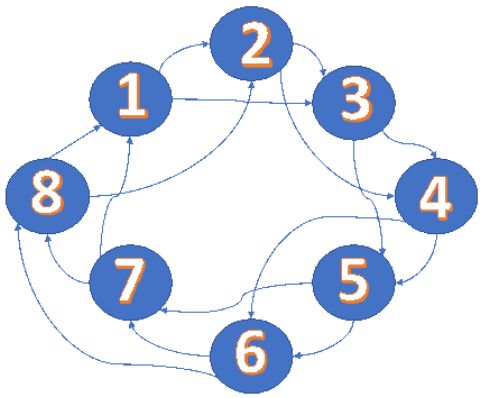


Figura 4.9: **Traccia esercizio 3.**

Per il corretto funzionamento della rete skip-ring è necessario che si verifichi una precisa condizione: non devono fallire due nodi adiacenti. Questo perchè, se questa condizione si verificasse, la rete non sarebbe più in grado di instradare i dati su un collegamento alternativo. Per poter enumerare tutte le configurazioni in cui falliscono almeno due nodi consecutivi, si è pensato di utilizzare e adattare la formula che permette di calcolare la Reliability di un sistema formato da N componenti che richiede almeno M componenti funzionanti per funzionare (M-out-of-N). La formula M-out-of-N è la seguente:

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

Tale formula va leggermente modificata andando a sottrarre al coefficiente binomiale tutte le combinazioni in cui il sistema non funziona.
Utilizzando la formula nel modo esatto, si vanno a sommare le probabilità che si verifichi una specifica configurazione di funzionamento del sistema in cui almeno M componenti funzionano. In particolare, ogni termine i-esimo della sommatoria tiene conto della probabilità che si verifichi una qualsiasi configurazione in cui funzionano N-i componenti, fino ad arrivare ad M, numero minimo di

componenti funzionanti richiesti dal sistema. Infatti, il coefficiente binomiale rappresenta il numero di combinazioni semplici di N elementi di classe i, ovvero il numero di configurazioni possibili in cui funzionano N-i componenti. Nel nostro caso, si può intuire che il numero minimo di componenti funzionanti per permettere al sistema di funzionare è 4, poiché per un numero di componenti funzionanti minore di 4 non è possibile trovare alcuna configurazione che verifichi la condizione di funzionamento del sistema; tale condizione, come detto in precedenza, è che non sia presente alcuna sequenza di almeno due nodi non funzionanti all'interno dell'anello. Quindi, facendo riferimento alla formula citata in precedenza, si imporrà M = 4 e N = 8.
L'adattamento sta nel fatto che di tutte le possibili configurazioni date da 4,5,6,7 e 8 nodi funzionanti, solo alcune garantiscono la condizione di funzionamento. Ciò implica che, per ogni termine della sommatoria, si sostituirà il coefficiente binomiale con il reale numero di configurazioni di N-i componenti funzionanti che garantiscono la condizione di funzionamento del sistema.
Il coefficiente binomiale rappresenterà l'upper bound del numero di configurazioni, al cui valore andare a sottrarre il numero di configurazioni che non garantiscono la condizione di funzionamento per N-i componenti funzionanti.
Di seguito l'analisi dei vari casi e quindi dei rispettivi termini della sommatoria. Indicheremo quindi, nella trattazione, con il colore rosso i nodi non funzionanti.

- **i=0**:

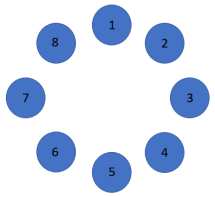


Figura 4.10: **Configurazione con 0 nodi non funzionanti.**

Il coefficiente binomiale 8 su 0 è uguale a 1 e complessivamente il termine della sommatoria nella formula originale è uguale a:

$$1R_m^8$$

Dunque, esiste ovviamente un'unica configurazione in cui tutti i componenti sono funzionanti e tale configurazione non viola la condizione di funzionamento del sistema, motivo per cui tale termine sarà considerato nella formula adattata.

- **i=1**:

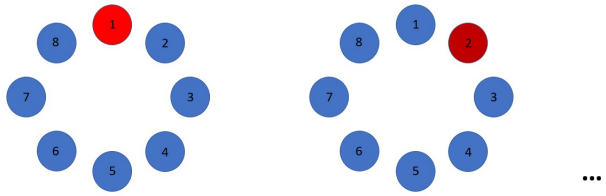


Figura 4.11: **Configurazione con 1 nodo non funzionante.**

Il coefficiente binomiale 8 su 1 è uguale a 8 e complessivamente il termine della sommatoria nella formula originale è uguale a:

$$8R_m^7(1 - R_m)$$

Dunque, esistono 8 configurazioni in cui 7 componenti sono funzionanti e di conseguenza uno è fallito e tali configurazioni non violano la condizione di funzionamento del sistema, motivo per cui tale termine sarà considerato nella formula adattata. Questo si può intuire dal fatto che condizione necessaria al fallimento del sistema è che vi siano almeno due nodi non funzionanti e in questo caso tale condizione non è rispettata. Per rappresentare tutte le configurazioni basta considerare le configurazioni in cui fallisce il nodo i-esimo e tutti gli altri nodi sono funzionanti.

- **i=2**:

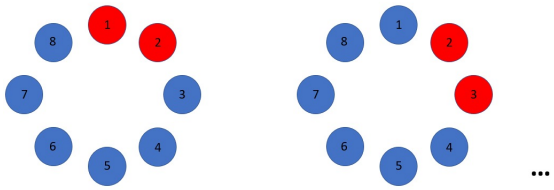


Figura 4.12: **Configurazione con 2 nodi non funzionanti.**

Il coefficiente binomiale 8 su 2 è uguale a 28 e complessivamente il termine della sommatoria nella formula originale è uguale a:

$$28R_m^6(1 - R_m)^2$$

Dunque, esistono 28 configurazioni in cui 6 componenti sono funzionanti e di conseguenza 2 sono falliti. Di queste configurazioni, solo quelle che prevedono i 2 componenti non funzionanti posti in modo consecutivo portano al fallimento del sistema. Per rappresentare ed enumerare queste configurazioni basta considerare le configurazioni in cui falliscono i nodi i-esimo e

i+1-esimo e tutti gli altri nodi sono funzionanti. Si può facilmente intuire che il numero di configurazioni con 2 nodi non funzionanti che portano al fallimento del sistema sono 8 delle 28 complessive, per cui il termine che sarà portato nella formula adattata è il seguente.

$$(28-8)R_m^6(1-R_m)^2 = 20R_m^6(1-R_m)^2$$

- **i=3**:

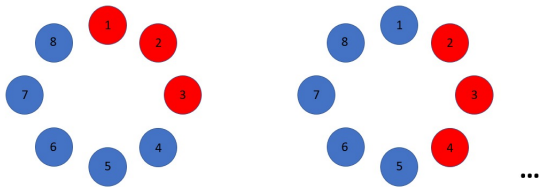


Figura 4.13: **Configurazione con 3 nodi non funzionanti.**

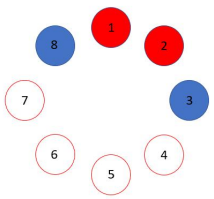
Il coefficiente binomiale 8 su 3 è uguale a 56 e complessivamente il termine della sommatoria nella formula originale è uguale a:

$$56R_m^5(1-R_m)^3$$

Dunque, esistono 56 configurazioni in cui 5 componenti sono funzionanti e di conseguenza 3 sono falliti. Di queste configurazioni, solo quelle che prevedono almeno 2 componenti non funzionanti posti in modo consecutivo portano al fallimento del sistema. Per rappresentare ed enumerare le configurazioni che portano al fallimento del sistema, si può osservare che si possono dividere in due partizioni:

1) Configurazioni in cui tutti i 3 componenti non funzionanti sono posti consecutivamente. Per rappresentare ed enumerare queste configurazioni basta considerare le configurazioni in cui falliscono i nodi i-esimo, i+1-esimo e i+2-esimo e tutti gli altri nodi sono funzionanti. Si può facilmente intuire che il numero di configurazioni appartenenti a questa partizione è 8.

2) Configurazioni in cui 2 dei 3 componenti non funzionanti sono posti consecutivamente. Per rappresentare ed enumerare queste configurazioni si può osservare la figura di seguito:



Nella configurazione rappresentata falliscono i nodi 1 e 2 evidenziati in rosso e uno solo dei quattro nodi bianchi con il contorno rosso. Una qualsiasi di queste quattro configurazioni appartiene alla partizione appena presentata. Per ottenere tutte le configurazioni appartenenti a questa partizione, si può generalizzare il ragionamento andando a considerare al posto dei nodi 1 e 2 i nodi i-esimo e i+1-esimo che falliscono e il fallimento di uno dei nodi i+3-esimo, i+4-esimo, i+5-esimo e i+6-esimo. In questo modo si ottiene il numero totale di configurazioni appartenenti a questa partizione, che è uguale a 8*4 = 32.

Dunque, il numero complessivo di configurazioni con 5 componenti funzionanti che porta al fallimento del sistema è dato dalla somma delle configurazioni appartenenti alle due partizioni discusse in precedenza, ovvero 8 + 32 = 40. Il termine che sarà portato nella formula adattata è il seguente.

$$(56 - 40)R_m^5(1 - R_m)^3 = 16R_m^5(1 - R_m)^3$$

- **i=4**:

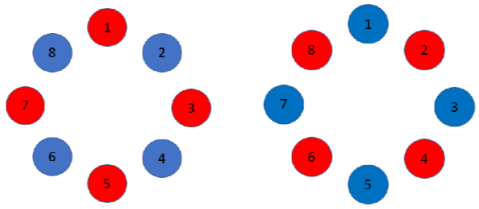


Figura 4.14: **Configurazioni con 4 nodi non funzionanti che non portano al fallimento.**

Il coefficiente binomiale 8 su 4 è uguale a 70 e complessivamente il termine della sommatoria nella formula originale è uguale a:

$$70R_m^4(1 - R_m)^4$$

Dunque, esistono 70 configurazioni in cui 4 componenti sono funzionanti e di conseguenza 4 sono falliti. Di queste configurazioni, solo quelle che prevedono un'alternanza tra componenti funzionanti e non funzionanti permettono il corretto funzionamento del sistema. Si può intuire che il numero di tali configurazioni è 2. Il termine che sarà portato nella formula adattata è il seguente:

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

Dunque, sommando tutti i termini calcolati, si ottiene l'espressione per la Reliability del sistema:

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

Dunque:

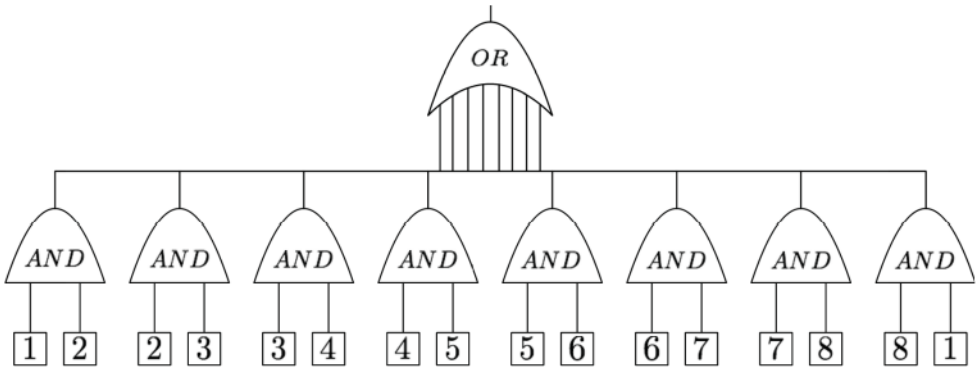
$$R_{sys} = 2R_m^4 + 8R_m^5 - 16R_m^5 + 82R_m^7 - R_m^8$$

Per rispondere al secondo quesito della traccia, bisogna trovare il valore di Reliability del sistema dopo 48h: supponendo che R_m segua una legge di fallimento esponenziale con tasso di fallimento pari a 0.005 per ora, bisogna sostituire nella formula precedente la seguente espressione per R_m:

$$R_m = e^{-0.005*48}$$

Il valore di Reliability che si ottiene è 0.729.

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:

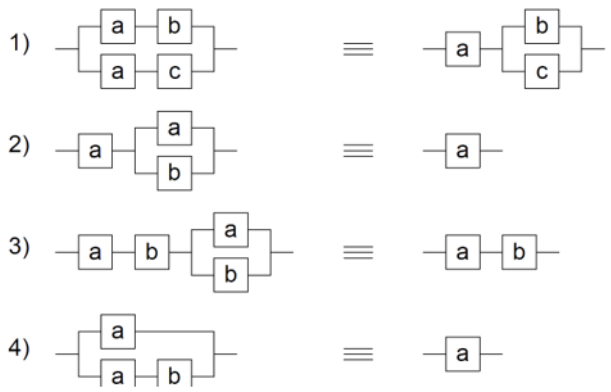


Si noti come il fault tree riportato presenta alcuni eventi ripetuti, ossia componenti contenuti in diverse foglie dell'albero; da cio ne consegue che i success path del problema non siano in realtà indipendenti. Se si intedesse calcolare la reliability del sistema a partire dal fault tree stesso, si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

l'utilizzo del conditioning.

4.4 Esercizio 4

Compare the reliability of the following schemes, assuming an exponential failure occurrence with following values:
$MTTF_A = 1000$ h
$MTTF_B = 9000$ h
$MMTF_C = 2000$ h



Assumendo gli MTTF forniti dalla traccia, le espressioni della Reliability dei componenti è la seguente,considerando come unità di misura per il tempo l'ora.
1) $R_A(t) = e^{\frac{-1}{1000}t}$
2) $R_B(t) = e^{\frac{-1}{9000}t}$
3) $R_C(t) = e^{\frac{-1}{2000}t}$

La configurazione di sinistra la denominiamo "1", la configurazione di destra la denominiamo "2".

- **Primo confronto** La reliability del sistema 1 può essere facilmente calcolata considerando il parallelo tra due serie, una composta da A-B e una da A-C. Di conseguenza otteniamo:

  $$R_{s1} = [1 - (1 - R_A R_B)(1 - R_A R_C)$$

  Per quanto riguarda invece il sistema 2, lo possiamo vedere come la serie tra A e un parallelo (B-C). Di conseguenza otteniamo:

  $$R_{s2} = R_A[1 - (1 - R_B)(1 - R_C)]$$

  Possiamo quindi, utilizzando le seguenti espressioni:

- $R_{s1}(t) = e^{-0.0026t} + e^{-0.0011t} + e^{-0.0015t}$
- $R_{s2}(t) = e^{-0.0016t} + e^{-0.0011t} + e^{-0.0015t}$

ricavare il confronto grafico tra le due configurazioni.

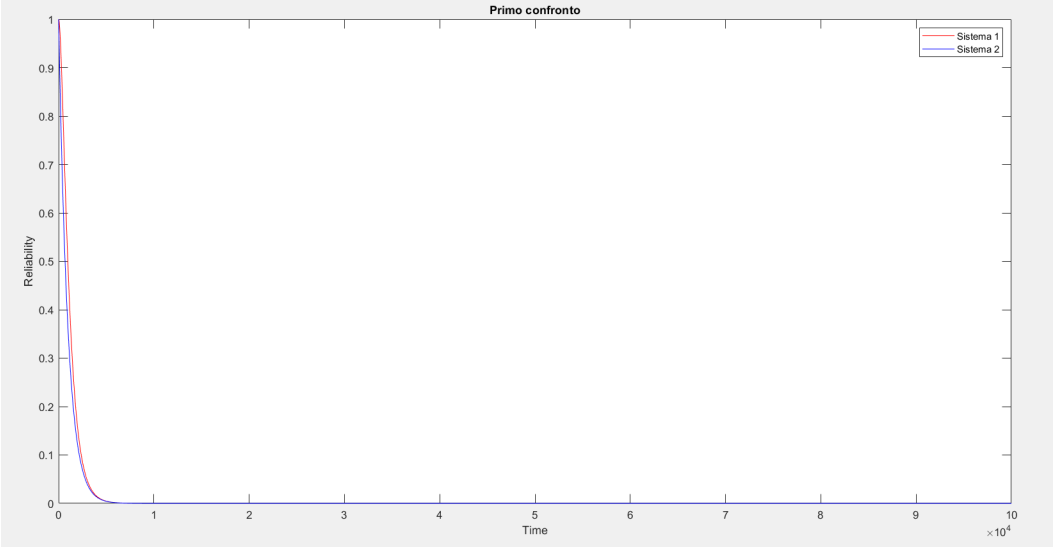


Figura 4.15: **Primo confronto.**

Notiamo come la curva del sistema 1 sia sempre al di sopra della curva del sistema 2, quindi possiamo concludere che lo schema 1 sia il migliore. Nella seconda configurazione i path di successo sono dipendenti tra loro, quindi il risultato era intuibile anche visivamente.

- **Secondo confronto** La reliability del sistema 1 può essere facilmente calcolata considerando una serie di due componenti, dove uno è semplicemente A e l'altro è il parallelo tra A-B. Di conseguenza otteniamo:

$$R_{s1} = R_A[]1 - (1 - R_A)(1 - R_B)]$$

Per quanto riguarda invece il sistema 2, è semplicemente A.

$$R_{s2} = R_A$$

Possiamo quindi, utilizzando le seguenti espressioni:

- $R_{s1}(t) = e^{-0.0021t} + e^{-0.002t} + e^{-0.001t}$
- $R_{s2}(t) = e^{-0.001t}$

ricavare il confronto grafico tra le due configurazioni.

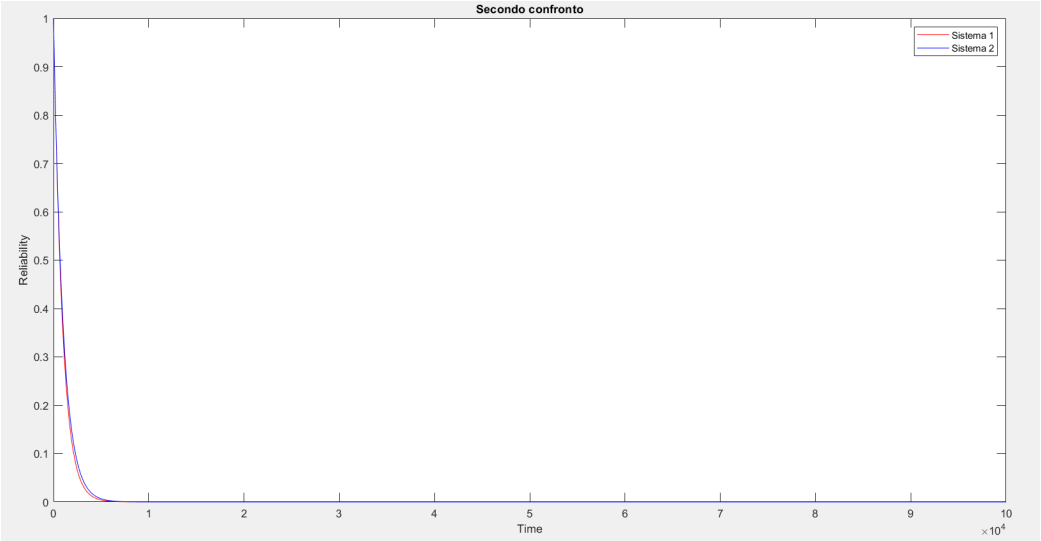


Figura 4.16: **Secondo confronto.**

Notiamo come la curva del sistema 2 sia sempre al di sopra della curva del sistema 1, quindi possiamo concludere che lo schema 2 sia il migliore. Il risultato era intuibile già dal fatto che nel secondo schema abbiamo un R_A, mentre nel primo moltiplichiamo R_A per una quantità minore di 1.

- **Terzo confronto** La reliability del sistema 1 può essere facilmente calcolata considerando una serie di tre componenti: il primo è A, il secondo è B, e il terzo è il parallelo tra A-B. Di conseguenza otteniamo:

$$R_{s1} = R_A R_B[1 - (1 - R_A)(1 - R_B)]$$

Per quanto riguarda invece il sistema 2, è semplicemente A.

$$R_{s2} = R_A R_B$$

Possiamo quindi, utilizzando le seguenti espressioni:

- $R_{s1}(t) = e^{-0.0022t} + e^{-0.0021t} + e^{-0.0012t}$
- $R_{s2}(t) = e^{-0.0011t}$

ricavare il confronto grafico tra le due configurazioni.

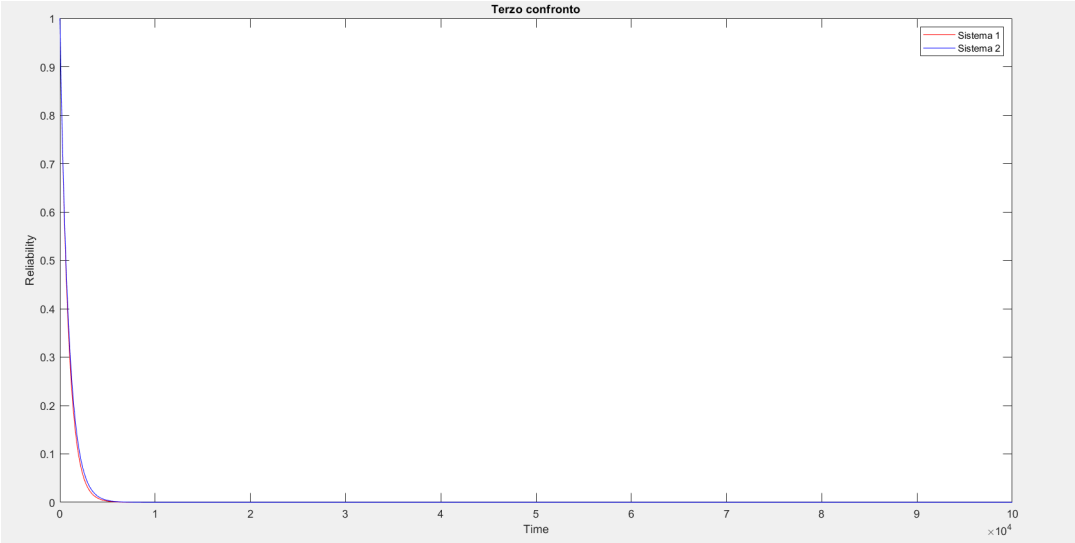


Figura 4.17: **Terzo confronto.**

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

- **Quarto confronto** La reliability del sistema 1 può essere facilmente calcolata considerando una parallelo di due componenti: il primo è A, il secondo è la serie A-B. Di conseguenza otteniamo:

$$R_{s1} = [1 - (1 - R_A)(1 - R_A R_B)]$$

Per quanto riguarda invece il sistema 2, è semplicemente A.

$$R_{s2} = R_A$$

Possiamo quindi, utilizzando le seguenti espressioni:

- $R_{s1}(t) = e^{-0.0021t} + e^{-0.0011t} + e^{-0.001t}$
- $R_{s2}(t) = e^{-0.001t}$

ricavare il confronto grafico tra le due configurazioni.

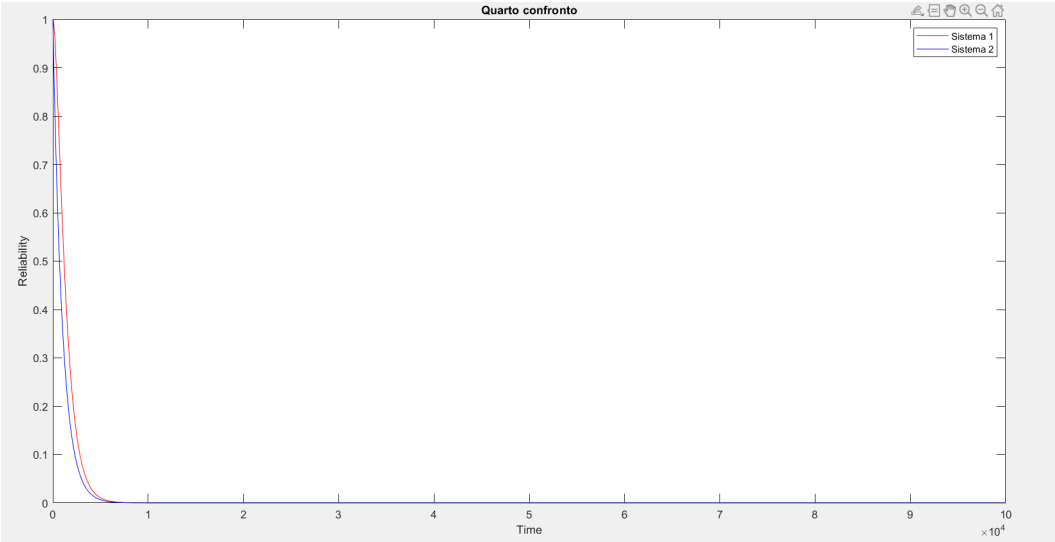
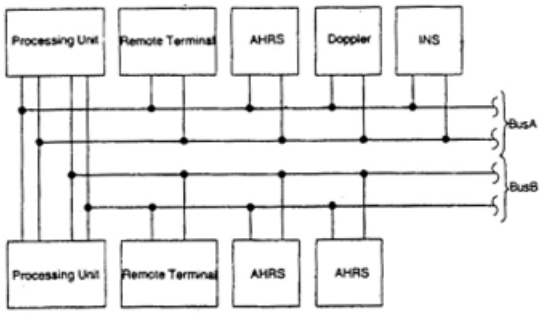


Figura 4.18: **Quarto confronto.**

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

4.5 Esercizio 5

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



a) Draw the reliability block diagram of the system.
b) Draw the Fault Tree of the system and analyze the minimal cutsets.
c) Calculate the reliability for a one-hour flight using the MTTF figures given in the table below. Assume that the exponential failure low applies and that the fault coverage is perfect.

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

d) Repeat (c), but this time, incorporate a coverage factor for the fault detection and reconfiguration of the processing units. Using the same failure data, determine the approximate fault coverage value that is required to obtain a reliability (at the end of one hour) of 0.99999.

Figura 4.19: **Traccia esercizio 5.**

Partiamo dalla risoluzione del **punto a**, che richiede di disegnare il reliability block diagram del sistema. 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.

Riportiamo il disegno risultante:

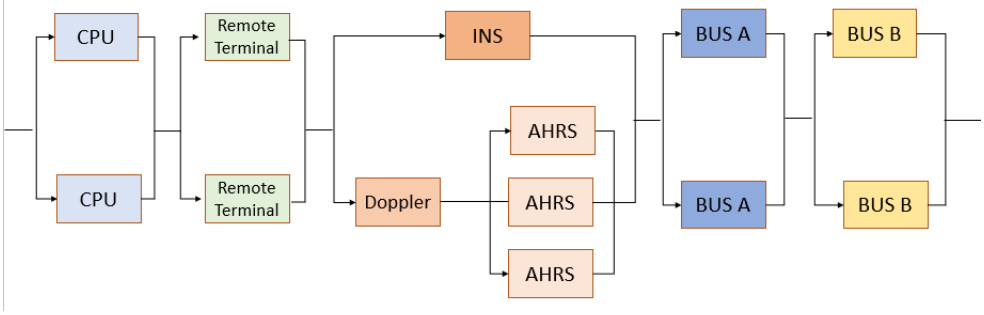


Figura 4.20: **punto a) RBD del sistema.**

Per quanto riguarda il **punto b**, è richiesto di sisegnare il fault tree del sistema e analizzare i cut set.
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:

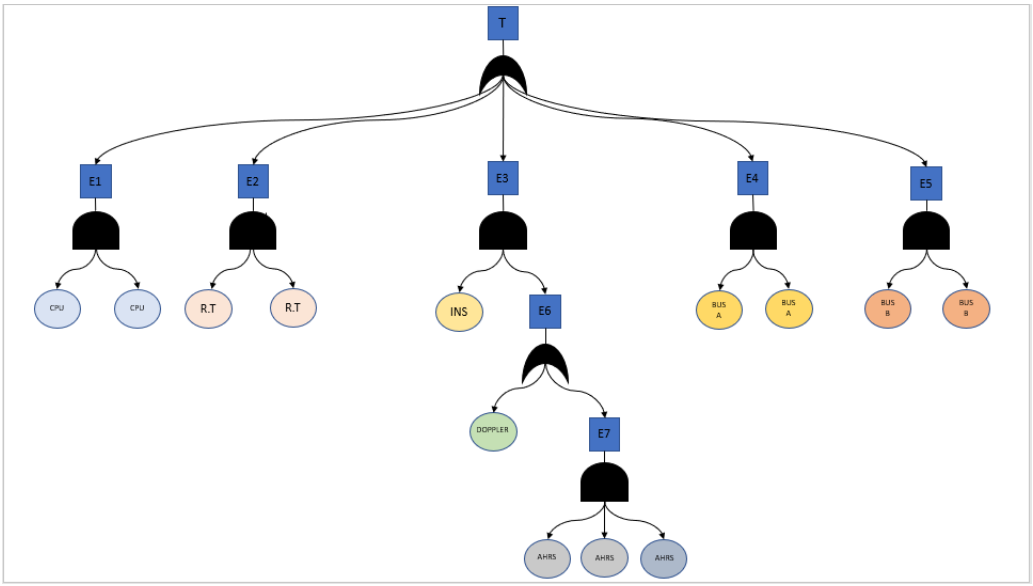


Figura 4.21: **punto b) Fault tree del sistema.**

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

- BUSA1*BUSA2
- BUSB1*BUSB2
- PU1*PU2
- RT1*RT2
- INS*DOPPL
- INS*AHRS1*AHRS2*AHRS3

Il **punto c**, invece prevede di calcolare la reliability per un volo di un'ora utilizzando i dati MTTF riportati nella tabella, supponendo che si applichi la legge di fallimento esponenziale e che la fault coverage sia perfetta.
Per farlo si ricava la seguente formula dell'RBD:

$$R_{sys} = (1-(1-R_{PU})^2)(1-(1-R_T)^2)(1-(1-R_{INS})^2)(1-(1-R_{BUSA})^2)(1-(1-R_{BUSB})^2)(1-(R_{DOPPL}(1-R_{AHRS})^3))$$

Considerando le seguenti quantità:
- $R_{PU}(t) = e^{\frac{-1}{10000}t}$
- $R_T(t) = e^{\frac{-1}{4500}t}$
- $R_{AHRS}(t) = e^{\frac{-1}{2000}t}$

- $R_{INS}(t) = e^{\frac{-1}{2000}t}$
- $R_{DOPPL}(t) = e^{\frac{-1}{500}t}$
- $R_{BUS}(t) = e^{\frac{-1}{60000}t}$

ricaviamo che $R_{sys}(1h) = 0.9999979$

Infine, per il **punto d** 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:

$$R_{proc} = R_{PU1} + c(1 - R_{PU1})R_{PU2}$$

Questa formula significa che: funziona il primo processore (R_{PU1}), oppure (il simbolo +), funziona il secondo processore dato che il primo non funziona con una certa probabilità c.
Sostituendo nella formula della reliability totale del sistema, otteniamo:

$R_{sys} = (R_{proc})(1 - (1 - R_T)^2)(1 - (1 - R_{INS})^2)(1 - (1 - R_{BUSA})^2)(1 - (1 - R_{BUSB})^2)(1 - (R_{DOPPL}(1 - R_{AHRS})^3))$

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: $c = 0.9$.

Capitolo 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).

Nell'effettuare la FFDA, si richiede di analizzare **MercuryErrorLog.txt** e **BGLErrorLog.txt** :
– grafico conteggio tuple + CWIN selezionato;
– raggruppare le voci con CWIN;
– distribuzioni empiriche;
– adattare l'affidabilità empirica (modello esponenziale + altri tentatividistribuzioni, ad esempio, weibull);
– analisi dell'adattamento tramite il test KS.

Condurre l'analisi per rispondere alle seguenti domande:
• le stesse finestre di coalescenza possono essere utilizzate in diversinodi (Mercury, BG/L) e categorie di errore (Mercury) ?
• per quanto riguarda il rapporto tra l'affidabilità del sistema e l'affidabilità dei nodi?
• esistono colli di bottiglia di affidabilità (ad es.contributori al numero totale di fallimenti)?
• fare nodi funzionali simili (es. due Mercury tg-cXnodi di calcolo o nodi BG/LI/O Ri:Mx:Nz) esibiscono parametri di affidabilità simili?
• esiste una relazione tra i tipi di errore e il nodo (Mercury)?

5.2 Risoluzione

Sono stati eseguiti i seguenti step per entrambi i super-calcolatori:
• analisi di sensitività del numero di tuple rispetto alla dimensione della finestra di coalescenza temporale
• costruzione delle tuple successiva alla scelta della finestra temporale da utilizzare
• analisi sommaria delle istanze di truncation
• analisi sommaria delle istanze di collision
• costruzione della distribuzione empirica della Reliability
• fitting della distribuzione empirica con una distribuzione nota
• verifica che la distribuzione ricavata empiricamente segua quella ricavata con il fitting, tramite K-S test.

Inoltre, per quanto riguarda Mercury abbiamo analizzato il nodo che fallisce maggiormente, abbiamo verificato se la dimensione della finestra scelta risultasse opportuna per tutti i componenti, abbiamo confrontato la reliability di ogni componente e fatto dunque delle considerazioni. Infine, abbiamo effettuato un confronto tra i due super-calcolatori. Per realizzare quanto detto si è fatto uso di Matlab e di tool ad esso connessi.

5.3 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**.

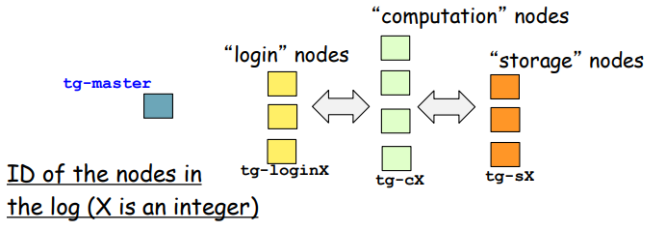


Figura 5.1: **Architettura Mercury.**

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.

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.

Un esempio di log:

```
1167667229 tg-c238 DEV
Component Info: Vendor Id =* *, Device Id =* *, Class Code =*
*, Seg Bus Dev Func =* * * * *
```

Figura 5.2: **Log Mercury.**

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:

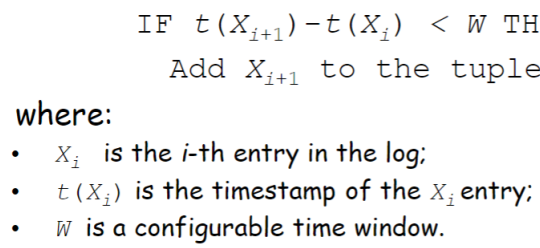


Figura 5.3: **Algoritmo di Tupling.**

5.3.1 Data manipolation - Dimensione Window

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.

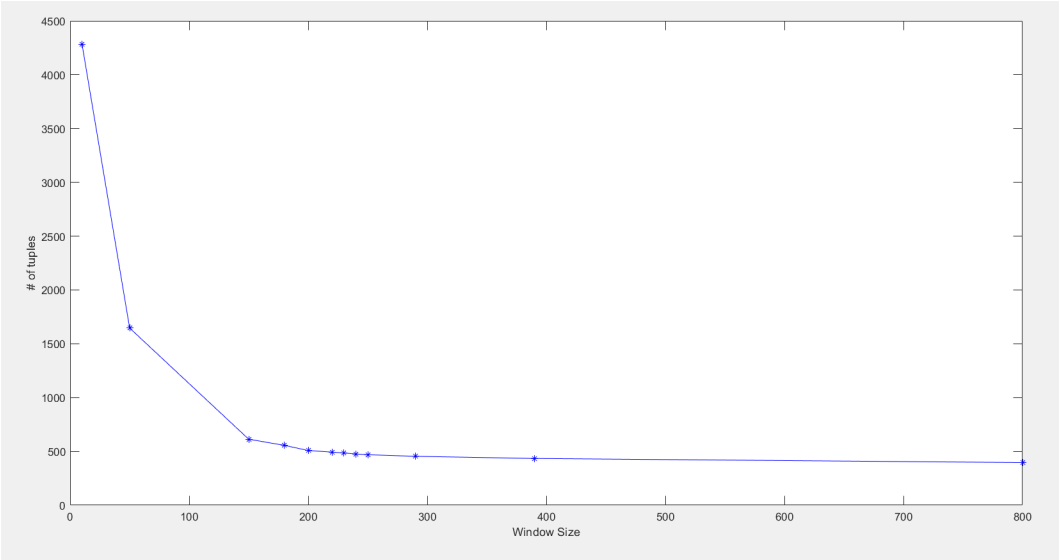


Figura 5.4: **Analisi di sensitività - Mercury.**

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.

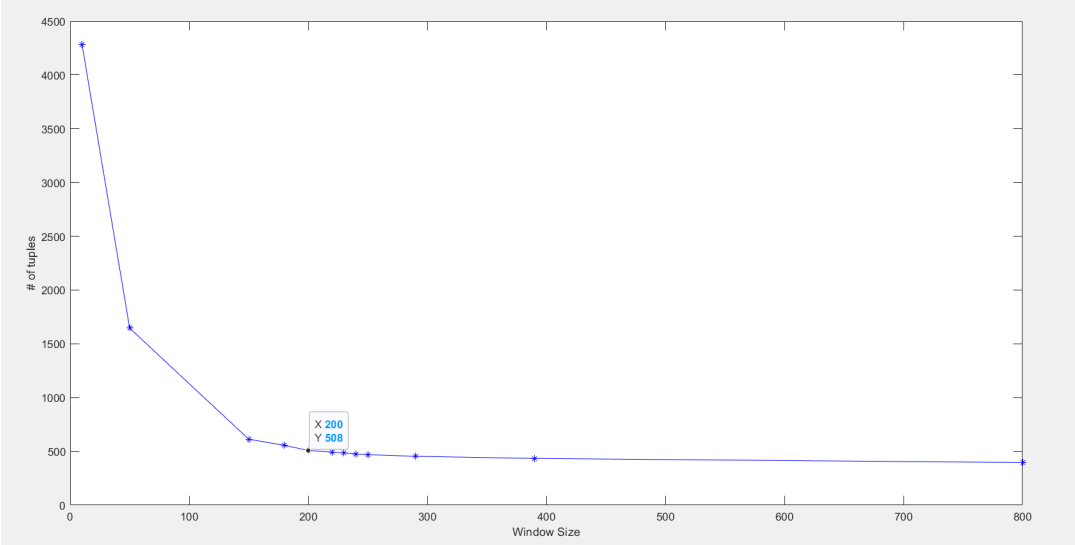


Figura 5.5: **Scelta window size - Mercury.**

Per tale motivo si è scelto una finestra temporale W = 200. Abbiamo dunque una finestra di 3 min e 20, e distinguiamo 508 tuple.

Abbiamo a questo punto utilizzato da bash il comando:
"$./tupling_{w}ith_{C}win.shMercuryErrorLog.txt200$".
In questo modo, accorpiamo entries nei log che hanno vicinanza spaziale. Più è veloce la creazione della tupla, più la tupla è piccola; altrimenti la tupla è grande.

ATTENZIONE, la distribuzione delle entries nelle tuple **NON** è uniforme: ci sono errori che si propagano, ed errori che non si propagano.

Viene dunque prodotto un file per ogni tupla, in cui troviamo i log che la compongono. Vengono prodotti anche altri 3 files:
1) *interarrivals.txt* : che contiene i tempi di arrivo di ogni tupla
2) *lenghts.txt* : che contiene la dimensione di ogni tupla
3) *startingPoints.txt* : che contiene i timestamp della prima entry della tupla.

Qual è l'effetto di aver scelto questa finestra temporale?
La finestra temporale mitiga le truncations e le collision:
- truncations: raggruppo entries relative allo stesso fallimento in tuple diverse (visione pessimistica)
- collision: raggruppo entries di fallimenti diversi nella stessa tupla (visione ottimista).

5.3.2 Data manipolation - Analisi delle truncations

Ci sono tuple, che contengono gli stessi log, la cui distanza è comparabile con la finestra temporale: possiamo avere il dubbio che siano delle **truncations**. Attraverso l'utilizzo di uno script python, riconosciamo 64 possibili truncations: ne analizziamo 2 nel dettaglio.

1) Tupla 436 e Tupla 437, nodo interessato tg-c880:

```
1174060462 tg-c880 PRO +BEGIN HARDWARE ERROR STATE AT CMC
1174060462 tg-c880 PRO +END HARDWARE ERROR STATE AT CMC
1174060462 tg-c880 PRO Device Error Info Section
1174060462 tg-c880 PRO Error Map: x
```

Figura 5.6: **Tupla 436.**

```
1174060802 tg-c880 PRO +BEGIN HARDWARE ERROR STATE AT CMC
1174060862 tg-c880 PRO +BEGIN HARDWARE ERROR STATE AT CMC
1174060862 tg-c880 PRO Device Error Info Section
```

Figura 5.7: **Tupla 437.**

In questo esempio si nota che in due tuple consecutive che distano tra loro di 82s, sono riportati errori della stessa tipologia e generati dallo stesso nodo c880.

Per 118 secondi le entries della tupla 437 non sono raggruppati insieme a quelle della tupla 436, questo è un esempio palese di truncation.

2) Tupla 92 e Tupla 93, nodo interessato tg-c027:

```
1168287345 tg-c027 I-O hda: packet command error: error=x
```

Figura 5.8: **Tupla 92.**

```
1168287585 tg-c027 NET connection down
1168287602 tg-c027 I-O hda: packet command error: error=x
```

Figura 5.9: **Tupla 93.**

In questo caso l'interpretazione non è immediata. Entrambe le entries sono generate dallo stesso nodo, tg-c027, ma la loro distanza temporale è pari ad 5004s, di gran lunga superiore alla finestra di coalescenza. A questo punto è necessario valutare se le due entries sono effettivamente scaturite da due errori della stessa tipologia, che si verificano in momenti differenti; ciò validerebbe il raggruppamento ottenuto attraverso il tupling standard; oppure, erroneamente, sono state separate.
I supercalcolatori, sono tipicamente utilizzati per eseguire processi di calcolo intensivi e il sistema fallisce se uno dei suoi job fallisce. Tali job consistono nell'esecuzione di routine ripetitive, se in un istante particolare dell'esecuzione di queste ultime si verifica un errore all'interno del sistema che impedisce alla routine di svolgere un particolare compito, questo stesso errore sarà riportato da tutte le routines successive fino al ripristino dello stato corretto del sistema.

5.3.3 Data manipolation - Analisi delle collisions

Per quanto riguarda le collisions, attraverso l'utilizzo di uno script in python riconosciamo 57 possibili collisions. Di queste ne analizziamo due:

1) Tupla 135:

```
1168579349 tg-c407 DEV +BEGIN HARDWARE ERROR STATE AT CPE
1168579349 tg-c407 DEV +Platform PCI Component Error Info Section
1168579349 tg-c407 DEV Component Info: Vendor Id =x x, Device Id =x x, Class Code =x x, Seg Bus Dev Func =x x x x x
1168579461 tg-c401 DEV +BEGIN HARDWARE ERROR STATE AT CPE
1168579461 tg-c401 DEV +Platform PCI Component Error Info Section
1168579461 tg-c401 DEV Component Info: Vendor Id =x x, Device Id =x x, Class Code =x x, Seg Bus Dev Func =x x x x x
```

Figura 5.10: **Tupla 135.**

Osservando nel dettaglio il messaggio contenuto nelle entries è possibile individuare relazioni tra di esse in quanto appartengono alla stessa categoria,ma sono registrate da due nodi separati.

E' possibile pensare allora, che il fallimento di uno dei due componenti, ha comportato il fallimento successivo: non possiamo affermare, che questo è un caso di collisione.

2) Tupla 77:

```
1168103794 tg-master NET setting node state to down
1168103794 tg-master NET setting node state to down
1168103794 tg-master NET setting node state to down
1168103794 tg-master NET setting node state to down
1168103794 tg-master NET setting node state to down
1168103794 tg-master NET setting node state to down
1168103795 tg-master NET setting node state to down
1168103797 tg-master NET setting node state to down
1168103822 tg-c735 DEV +BEGIN HARDWARE ERROR STATE AT CPE
1168103822 tg-c735 DEV +Platform PCI Component Error Info Section
1168103822 tg-c735 DEV Component Info: Vendor Id =x x, Device Id =x x, Class Code =x x, Seg Bus Dev Func =x x x x x
1168103822 tg-c735 DEV Component Info: Vendor Id =x x, Device Id =x x, Class Code =x x, Seg Bus Dev Func =x x x x x
1168103914 tg-c696 I-O error, dev x:x (hda), sector x
1168103914 tg-c696 I-O error, dev x:x (hda), sector x
1168103914 tg-c696 I-O error, dev x:x (hda), sector x
1168103914 tg-c696 I-O error, dev x:x (hda), sector x
1168103914 tg-c696 I-O error, dev x:x (hda), sector x
1168103914 tg-c696 I-O error, dev x:x (hda), sector x
1168103914 tg-c696 I-O error, dev x:x (hda), sector x
1168103914 tg-c696 I-O error, dev x:x (hda), sector x
1168103914 tg-c696 I-O error, dev x:x (hda), sector x
1168103914 tg-c696 I-O error, dev x:x (hda), sector x
1168103914 tg-c696 I-O error, dev x:x (hda), sector x
1168103914 tg-c696 I-O error, dev x:x (hda), sector x
1168103914 tg-c696 I-O error, dev x:x (hda), sector x
1168103914 tg-c696 I-O error, dev x:x (hda), sector x
1168103914 tg-c696 I-O error, dev x:x (hda), sector x
1168103914 tg-c696 I-O error, dev x:x (hda), sector x
1168103914 tg-c696 I-O error, dev x:x (hda), sector x
1168103914 tg-c696 I-O error, dev x:x (hda), sector x
1168103914 tg-c696 I-O error, dev x:x (hda), sector x
1168103914 tg-c696 I-O error, dev x:x (hda), sector x
1168103914 tg-c696 I-O error, dev x:x (hda), sector x
1168103914 tg-c696 I-O error, dev x:x (hda), sector x
1168103985 tg-master NET setting node state to down
1168104044 tg-master NET setting node state to down
1168104044 tg-master NET setting node state to down
1168104044 tg-master NET setting node state to down
1168104044 tg-master NET setting node state to down
1168104044 tg-master NET setting node state to down
1168104044 tg-master NET setting node state to down
```

Figura 5.11: **Tupla 77.**

Osservando nel dettaglio il messaggio contenuto nelle entries non è possibile individuare alcuna relazione tra di esse in quanto appartengono a due categorie differenti ed inoltre sono registrate da due nodi separati. Da questa analisi è ragionevole affermare che si tratta di una collision in quanto non esiste alcuna correlazione tra gli eventi registrati nelle due entries.

5.3.4 Data analysis

L'ultimo passo del processo di FFDA consiste nell'analisi statistica dei dati per poterne estrapolare 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 uno script in Matlab, fornitoci nelle slides presenti nel materiale didattico, che ci ha fornito i seguenti risultati:

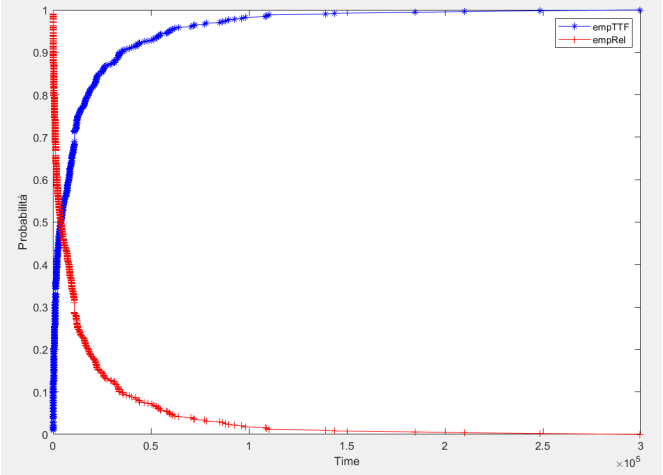


Figura 5.12: **Grafico Reliability - Mercury.**

Al fine di verificare quale delle distribuzioni note è in grado di approssimare al meglio le funzioni di probabilità ottenute, si possono utilizzare due tool diversi di Matlab: Curve Fitting, Distribution Fitting. Il Distibution Fitting da il fitting della distribuzione, il Curve Fitting invece da un modello statistico.
Per quanto riguarda l'utilizzo del Curve Fitting, vediamo:
1) Esponenziale ad 1 parametro:

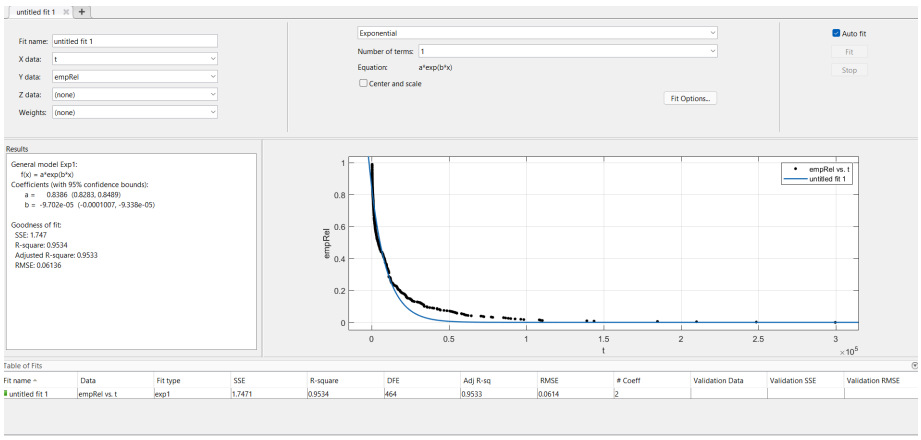


Figura 5.13: **Esponenziale ad un parametro - Mercury.**

2) Esponenziale ad 2 parametri:

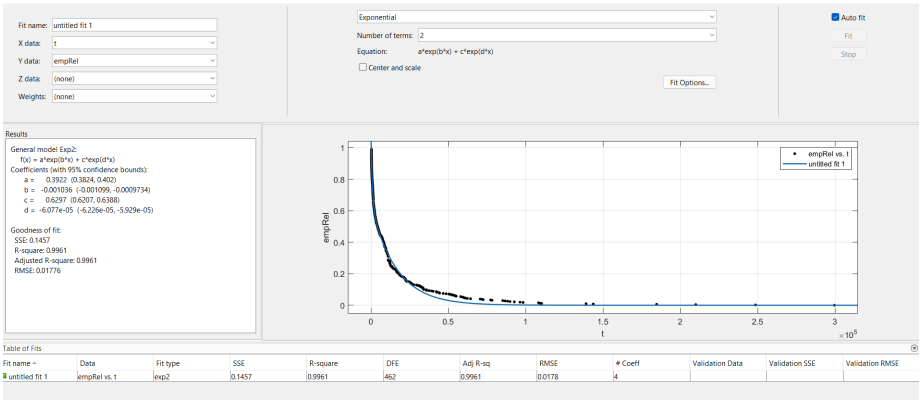


Figura 5.14: **Esponenziale a due parametri - Mercury.**

Confrontando i due tipi di fitting, vediamo come l'esponenziale a due parametri rappresenta molto meglio l'andamento che avevamo ottenuto (questo lo si può notare anche dall'SSE che nella prima figura è più elevato di 1, al contrario, nel secondo caso è quasi pari a 0).
Poichè l'andamento lo rappresentiamo in modo esponenziale, si deduce che c'è stato il fallimento di un componente hardware.
Per verificare che effettivamente i dati ricavati empiricamente seguissero la distribuzione esponenziale, si è utilizzato il test di Kolmogorov-Smirnov.

```
>> [H,P,K]=kstest2(empRel,thRel1(t))

H =

  logical

   0


P =

    0.6666


K =

    0.0472
```

Figura 5.15: **K-S test - Mercury.**

Dalla figura 5.15 vediamo come l'ipotesi che il nostro andamento è ben rappresentato dall'esponenziale a due parametri viene accettata (in quanto H=0), con un p-value molto elevato.

Abbiamo voluto vedere se avessimo avuto con l'utilizzo del tool Distribution Fitting, gli stessi risultati.

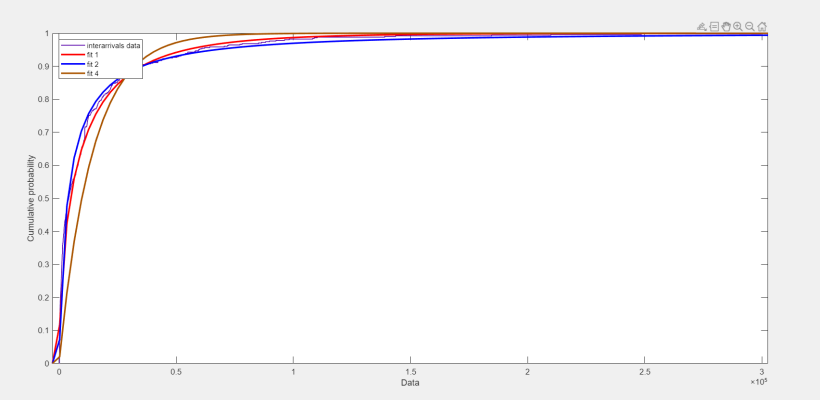


Figura 5.16: **Distribution Fitting - Mercury.**

Nell'immagine vediamo:
- interarrivals (l'andamento degli arrivi);
- fit 1 (andamento Esponenziale);
- fit 2 (andamento Lognormale);
- fit 3 (andamento Weibull).

Effettuando uno zoom della figura 5.16 possiamo vedere:

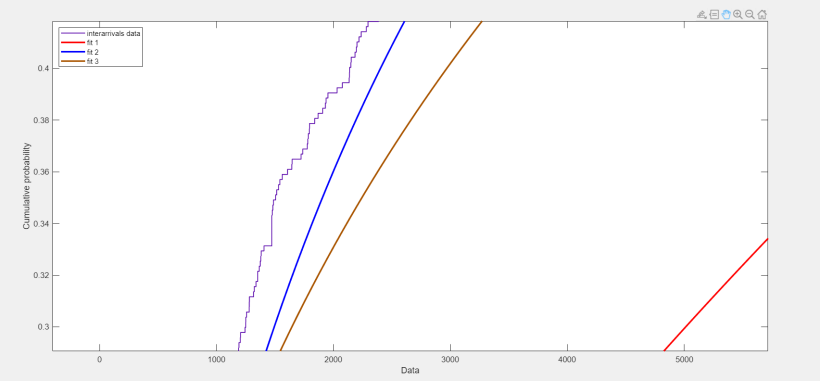


Figura 5.17: **Zoom Distribution Fitting - Mercury.**

Probabilmente, l'andamento lognormale (blu) è quello che si avvicina maggiormente al nostro: dobbiamo fare attenzione, perchè attraverso l'utilizzo del tool Distribution Fitting, stiamo dicendo che effettuando il fitting dei dati ci

avviciniamo al grafico di una lognormale, ma non stiamo predicendo nessun andamento futuro, ovvero non stiamo rappresentando un modello.

5.3.5 Statistiche

Attraverso l'utilizzo dello script *filter.sh* possiamo studiare alcune caratteristiche del supercalcolatore. Applicandolo al nostro insieme di log di Mercury otteniamo:

```
== Total error entries ==
80854

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

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

Figura 5.18: **Statistiche - Mercury.**

Da questa immagine si evincono i nodi che falliscono maggiormente e quanti errori contengono, e per ogni componente del supercalcolatore quanti errori si sono verificati.

5.3.6 Statistiche - Analisi del nodo critico

Il nodo critico è senza dubbio tg-c401, come è emerso anche dal filtraggio rappresentato in figura 5.18.
Si è proceduto quindi a realizzare la sensitivity analysis per tale nodo: il valore di finestra di coalescenza più appropriato è probabilmente 180 (3 min), otteniamo 72 tuple.

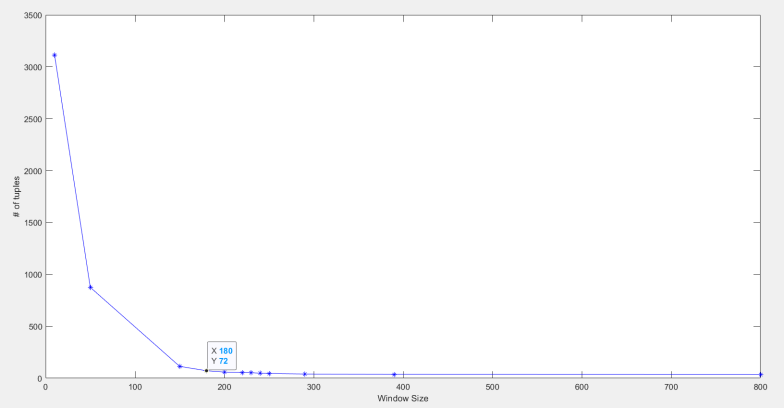


Figura 5.19: **Analisi sensitività del nodo critico tg-c401 - Mercury.**

Di seguito si riporta la reliability del nodo stesso:

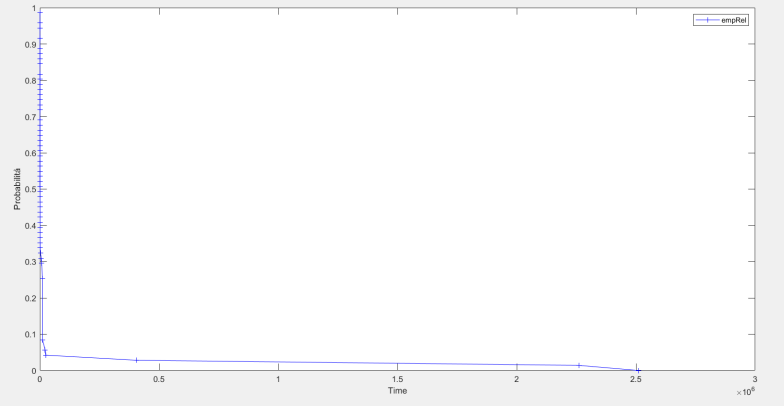


Figura 5.20: **Reliability del nodo critico tg-c401 - Mercury.**

Poichè analizzando gli errori di tg-c401, abbiamo visto che sono tutti riferiti ai componenti DEV e MEM, abbiamo effettuato l'analisi di sensitività in modo separato per entrambi gli errori, per vedere se avessero bisogno di finestre di dimensioni differenti per realizzare un tupling efficiente.

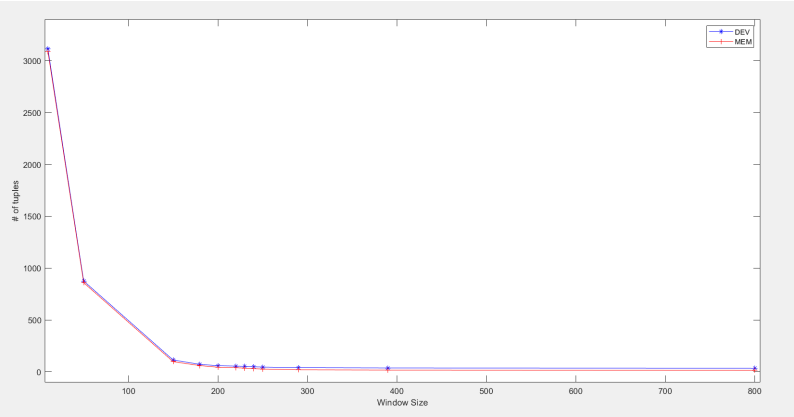


Figura 5.21: **Analisi sensitività errori tg-c401 - Mercury.**

Dall'immagine si evince che per entrambi gli errori una Window Size pari a 180 sia adeguata.

Abbiamo inoltre confrontato due nodi dello stesso tipo:in particolare abbiamo confrontato il nodo tg-c401 con il nodo tg-c238. Pur essendo i nodi della stessa natura, sono caratterizzati da errori differenti; infatti, effettuando il filtering degli errori di tg-c238, vediamo oltre a DEV e MEM, fallimenti riguardanti NET. Dall'analisi di sensitività del nodo tg-c238 ne ricaviamo che la Window Size per esso risulta essere pari a 220. Riportiamo il grafico che la rappresenta:

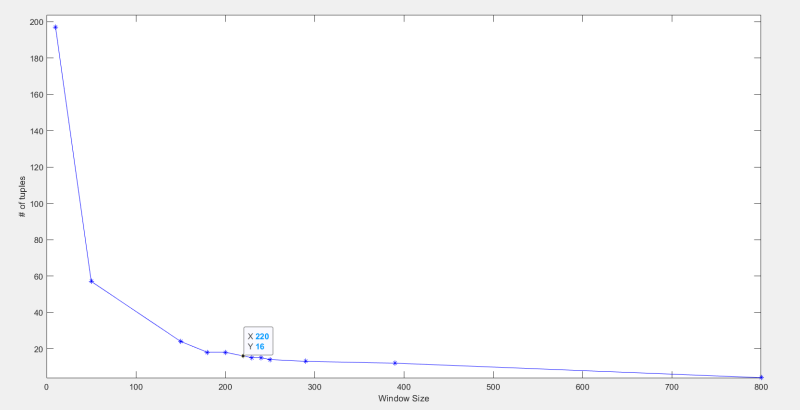


Figura 5.22: **Analisi sensitività errori tg-c238 - Mercury.**

Possiamo dunque, come controprova, verificare che la reliability del nodo tg-c238 risulta essere migliore di quella tg-c401, in quanto quest'ultimo rappresenta il collo di bottiglia del supercalcolatore in esame.

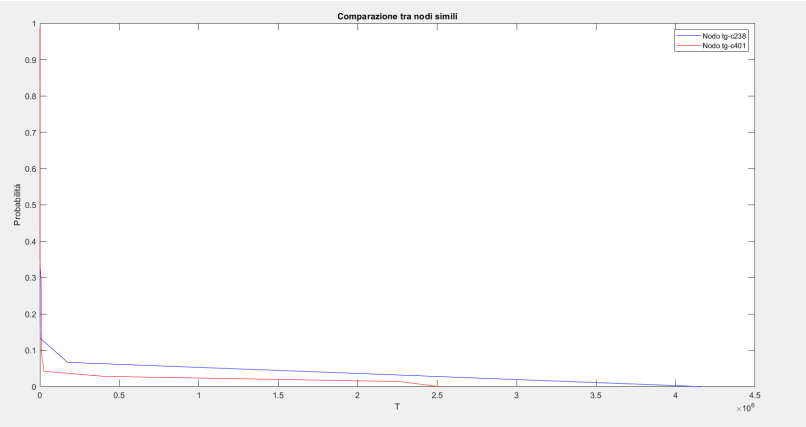


Figura 5.23: **Confronto nodi simili - Mercury.**

5.3.7 Statistiche - Analisi dei componenti

Successivamente ci siamo impegnate nel capire se la Window Size imposta durante l'analisi di sensitività pari a 200, fosse adeguata per tutti i componenti del supercalcolatore. Di conseguenza, abbiamo filtrato gli errori per ogni componente e li abbiamo plottati ricavando le seguenti misure:

1) DEV: finestra 200, 334 tuple
2) MEM: finestra 200, 105 tuple
3) I/O: finestra 200, 96 tuple
4) PRO: finestra 180, 70 tuple
5) NET: finestra 220, 66 tuple
6) OTH: finestra 150, 13 tuple

Da questi risultati capiamo che, per i componenti che falliscono con una frequenza maggiore, la dimensione della finestra risulta essere corretta, poichè probabilmente nell'analisi effettuata nei paragrafi precedenti, il loro contributo risultava essere predominante; ma, se andiamo ad analizzare i singoli componenti, vediamo come per PRO o NET, sarebbe necessaria una finestra di durata inferiore affinché non si verifichino problemi di collisione.

Avendo a disposizione i dati filtrati per ogni componente, abbiamo plottato la reliability di ognuno di essi ed ottenuto il seguente grafico:

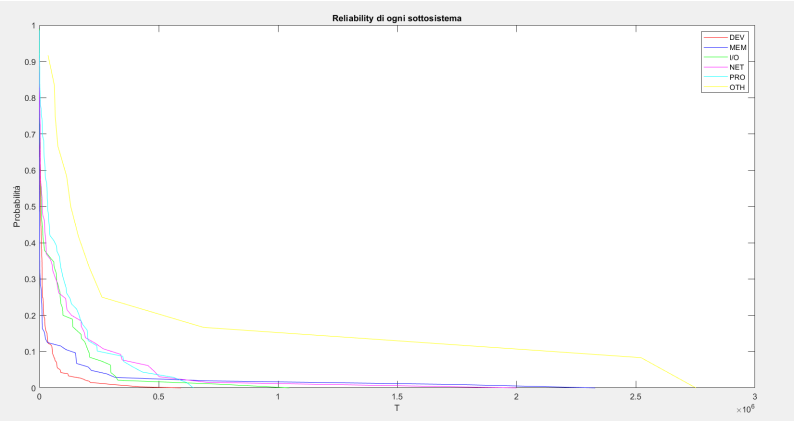


Figura 5.24: **Reliability dei sottosistemi - Mercury.**

Dal grafico si evince che effettivamente il componente che ha una reliability minore (e quindi fallisce con maggiore probabilità) è DEV: abbiamo dimostrato che il filtering effettuato precedentemente risulta essere corretto.

5.4 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.

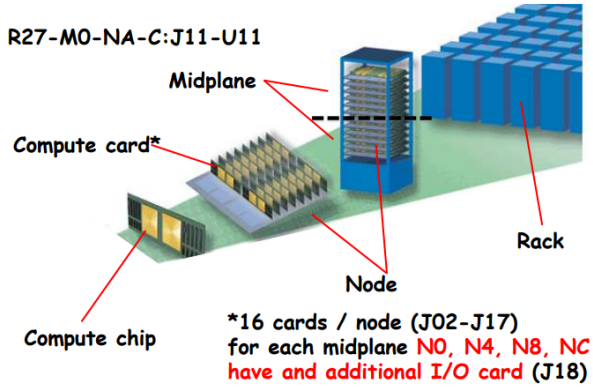


Figura 5.25: **Architettura BlueGene/L.**

Essendo dinanzi a file di log già collezionati e opportunamente filtrati, la FFDA in talcaso 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.

Un esempio di log:

```
1128621351 R04-M0-N0 J18-U01
Lustre mount FAILED : bglio66 : block_id : location
```

Figura 5.26: **Log BlueGene/L.**

5.4.1 Data manipolation - Sensitivity Analysis

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.

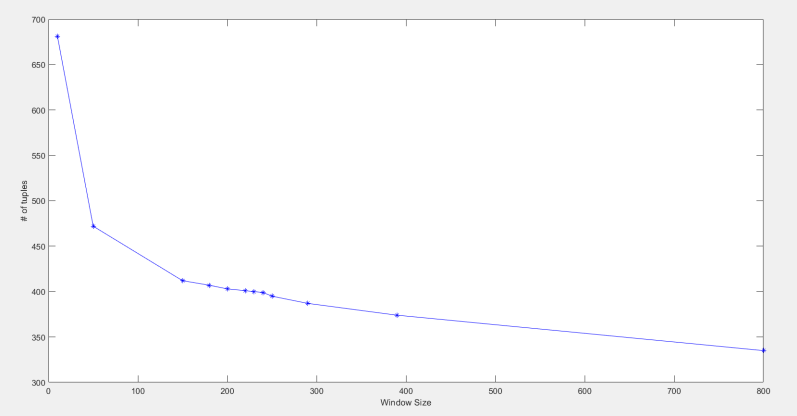


Figura 5.27: **Analisi di sensitività - BlueGene/L.**

Si è scelto una finestra temporale W = 180. Abbiamo dunque una finestra di 3 min, e distinguiamo 407 tuple.

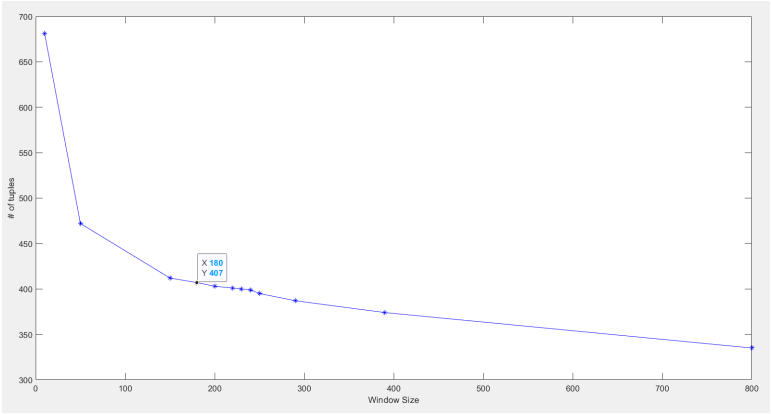


Figura 5.28: **Window Size - BlueGene/L.**

Abbiamo a questo punto utilizzato da bash il comando:
"$./tupling_{w}ith_{C}win.shBGLErrorLog.txt180$".
In questo modo, accorpiamo entries nei log che hanno vicinanza spaziale. Più è veloce la creazione della tupla, più la tupla è piccola; altrimenti la tupla è grande.
Attenzione, la distribuzione delle entries nelle tuple **NON** è uniforme: ci sono errori che si propagano, ed errori che non si propagano.
Viene dunque prodotto un file per ogni tupla, in cui troviamo i log che la compongono. Vengono prodotti anche altri 3 files:
1) *interarrivals.txt* : che contiene i tempi di arrivo di ogni tupla
2) *lenghts.txt* : che contiene la dimensione di ogni tupla
3) *startingPoints.txt* : che contiene i timestamp della prima entry della tupla.

5.4.2 Data manipolation - Analisi delle truncations

Qual è l'effetto di aver scelto questa finestra temporale?
La finestra temporale mitiga le truncations e le collision:
- truncations: raggruppo entries relative allo stesso fallimento in tuple diverse (visione pessimistica)
- collision: raggruppo entries di fallimenti diversi nella stessa tupla (visione ottimista).
Ci sono tuple, che contengono gli stessi log, la cui distanza è comparabile con la finestra temporale: possiamo avere il dubbio che siano delle **truncations**. Analizziamo due tuple nel dettaglio per capire se rappresentano truncations o meno.

1) Tupla 154 e Tupla 155, nodo interessato R63-M0-N0:

```
1131491322 R63-M0-N0 J18-U11 ciod: Error loading /p/gb1/bmiller/CALTECH/test/mdcask.opt: program image too big, 1380033264 > 532152320
1131491322 R63-M0-N0 J18-U01 ciod: Error loading /p/gb1/bmiller/CALTECH/test/mdcask.opt: program image too big, 1380033264 > 532152320
1131491322 R63-M0-NC J18-U11 ciod: Error loading /p/gb1/bmiller/CALTECH/test/mdcask.opt: program image too big, 1380033264 > 532152320
1131491322 R63-M0-NC J18-U01 ciod: Error loading /p/gb1/bmiller/CALTECH/test/mdcask.opt: program image too big, 1380033264 > 532152320
1131491322 R63-M0-N8 J18-U11 ciod: Error loading /p/gb1/bmiller/CALTECH/test/mdcask.opt: program image too big, 1380033264 > 532152320
1131491322 R63-M0-N8 J18-U01 ciod: Error loading /p/gb1/bmiller/CALTECH/test/mdcask.opt: program image too big, 1380033264 > 532152320
1131491322 R63-M0-N4 J18-U11 ciod: Error loading /p/gb1/bmiller/CALTECH/test/mdcask.opt: program image too big, 1380033264 > 532152320
1131491322 R63-M0-N4 J18-U01 ciod: Error loading /p/gb1/bmiller/CALTECH/test/mdcask.opt: program image too big, 1380033264 > 532152320
1131491350 R62-M0-NC J18-U11 ciod: Error loading /p/gb1/bmiller/CALTECH/test/mdcask.opt: program image too big, 1380033264 > 532152320
1131491350 R62-M0-NC J18-U01 ciod: Error loading /p/gb1/bmiller/CALTECH/test/mdcask.opt: program image too big, 1380033264 > 532152320
1131491350 R62-M0-N0 J18-U11 ciod: Error loading /p/gb1/bmiller/CALTECH/test/mdcask.opt: program image too big, 1380033264 > 532152320
1131491350 R62-M0-N0 J18-U01 ciod: Error loading /p/gb1/bmiller/CALTECH/test/mdcask.opt: program image too big, 1380033264 > 532152320
1131491350 R62-M0-N4 J18-U11 ciod: Error loading /p/gb1/bmiller/CALTECH/test/mdcask.opt: program image too big, 1380033264 > 532152320
1131491350 R62-M0-N4 J18-U01 ciod: Error loading /p/gb1/bmiller/CALTECH/test/mdcask.opt: program image too big, 1380033264 > 532152320
1131491350 R62-M0-N8 J18-U11 ciod: Error loading /p/gb1/bmiller/CALTECH/test/mdcask.opt: program image too big, 1380033264 > 532152320
1131491350 R62-M0-N8 J18-U01 ciod: Error loading /p/gb1/bmiller/CALTECH/test/mdcask.opt: program image too big, 1380033264 > 532152320
```

Figura 5.29: **Tupla 154.**

```
1131491644 R63-M0-N0 J18-U11 ciod: Error loading /p/gb1/bmiller/CALTECH/test/mdcask.opt: program image too big, 1378448112 > 532152320
1131491644 R63-M0-N0 J18-U01 ciod: Error loading /p/gb1/bmiller/CALTECH/test/mdcask.opt: program image too big, 1378448112 > 532152320
1131491644 R63-M0-NC J18-U11 ciod: Error loading /p/gb1/bmiller/CALTECH/test/mdcask.opt: program image too big, 1378448112 > 532152320
1131491644 R63-M0-NC J18-U01 ciod: Error loading /p/gb1/bmiller/CALTECH/test/mdcask.opt: program image too big, 1378448112 > 532152320
1131491644 R63-M0-N8 J18-U11 ciod: Error loading /p/gb1/bmiller/CALTECH/test/mdcask.opt: program image too big, 1378448112 > 532152320
1131491644 R63-M0-N8 J18-U01 ciod: Error loading /p/gb1/bmiller/CALTECH/test/mdcask.opt: program image too big, 1378448112 > 532152320
1131491644 R63-M0-N4 J18-U11 ciod: Error loading /p/gb1/bmiller/CALTECH/test/mdcask.opt: program image too big, 1378448112 > 532152320
1131491644 R63-M0-N4 J18-U01 ciod: Error loading /p/gb1/bmiller/CALTECH/test/mdcask.opt: program image too big, 1378448112 > 532152320
```

Figura 5.30: **Tupla 155.**

In questo esempio si nota che in due tuple consecutive che distano tra loro di 56s, sono riportati errori della stessa tipologia e generati dallo stesso nodo R63-M0-N0. In questo esempio: per 124 secondi le entries della tupla 155 non sono raggruppati insieme a quelle della tupla 154, questo è un esempio palese di truncation.

2) Tupla 123 e Tupla 124, nodo interessato R63-M0-N0:

```
1131045342 R63-M0-N0 J18-U11 ciod: Error loading /home/antypas1/bgl_F3/Sod3d64IO/flash3: invalid or missing program image, No such file or directory
1131045342 R63-M0-N0 J18-U01 ciod: Error loading /home/antypas1/bgl_F3/Sod3d64IO/flash3: invalid or missing program image, No such file or directory
1131045342 R63-M0-NC J18-U11 ciod: Error loading /home/antypas1/bgl_F3/Sod3d64IO/flash3: invalid or missing program image, No such file or directory
1131045342 R63-M0-NC J18-U01 ciod: Error loading /home/antypas1/bgl_F3/Sod3d64IO/flash3: invalid or missing program image, No such file or directory
1131045342 R63-M0-N8 J18-U11 ciod: Error loading /home/antypas1/bgl_F3/Sod3d64IO/flash3: invalid or missing program image, No such file or directory
1131045342 R63-M0-N8 J18-U01 ciod: Error loading /home/antypas1/bgl_F3/Sod3d64IO/flash3: invalid or missing program image, No such file or directory
1131045342 R63-M0-N4 J18-U11 ciod: Error loading /home/antypas1/bgl_F3/Sod3d64IO/flash3: invalid or missing program image, No such file or directory
1131045342 R63-M0-N4 J18-U01 ciod: Error loading /home/antypas1/bgl_F3/Sod3d64IO/flash3: invalid or missing program image, No such file or directory
```

Figura 5.31: **Tupla 123.**

```
1131045587 R63-M0-N0 J18-U11 ciod: Error loading /home/antypas1/bgl_F3/FLASH3/Sod3d64IO/flash3: invalid or missing program image, No such file or directory
1131045587 R63-M0-N0 J18-U01 ciod: Error loading /home/antypas1/bgl_F3/FLASH3/Sod3d64IO/flash3: invalid or missing program image, No such file or directory
1131045587 R63-M0-NC J18-U11 ciod: Error loading /home/antypas1/bgl_F3/FLASH3/Sod3d64IO/flash3: invalid or missing program image, No such file or directory
1131045587 R63-M0-NC J18-U01 ciod: Error loading /home/antypas1/bgl_F3/FLASH3/Sod3d64IO/flash3: invalid or missing program image, No such file or directory
1131045587 R63-M0-N8 J18-U11 ciod: Error loading /home/antypas1/bgl_F3/FLASH3/Sod3d64IO/flash3: invalid or missing program image, No such file or directory
1131045587 R63-M0-N8 J18-U01 ciod: Error loading /home/antypas1/bgl_F3/FLASH3/Sod3d64IO/flash3: invalid or missing program image, No such file or directory
1131045587 R63-M0-N4 J18-U11 ciod: Error loading /home/antypas1/bgl_F3/FLASH3/Sod3d64IO/flash3: invalid or missing program image, No such file or directory
1131045587 R63-M0-N4 J18-U01 ciod: Error loading /home/antypas1/bgl_F3/FLASH3/Sod3d64IO/flash3: invalid or missing program image, No such file or directory
```

Figura 5.32: **Tupla 124.**

In questo caso l'interpretazione non è immediata. Entrambe le entries sono generate dallo stesso nodo, R63-M0-N0, ma la loro distanza temporale è pari ad 2936s, di gran lunga superiore alla finestra di coalescenza. A questo punto è necessario valutare se le due entries sono effettivamente scaturite da due errori della stessa tipologia, che si verificano in momenti differenti; ciò validerebbe il raggruppamento ottenuto attraverso il tupling standard; oppure, erroneamente,

sono state separate. I supercalcolatori, sono tipicamente utilizzati per eseguire processi di calcolo intensivi e il sistema fallisce se uno dei suoi job fallisce. Tali job consistono nell'esecuzione di routine ripetitive, se in un istante particolare dell'esecuzione di queste ultime si verifica un errore all'interno del sistema che impedisce alla routine di svolgere un particolare compito, questo stesso errore sarà riportato da tutte le routines successive fino al ripristino dello stato corretto del sistema.

5.4.3 Data manipolation - Analisi delle collisions

Per quanto riguarda le collisions, ne analizziamo due:

1) Tupla 363:

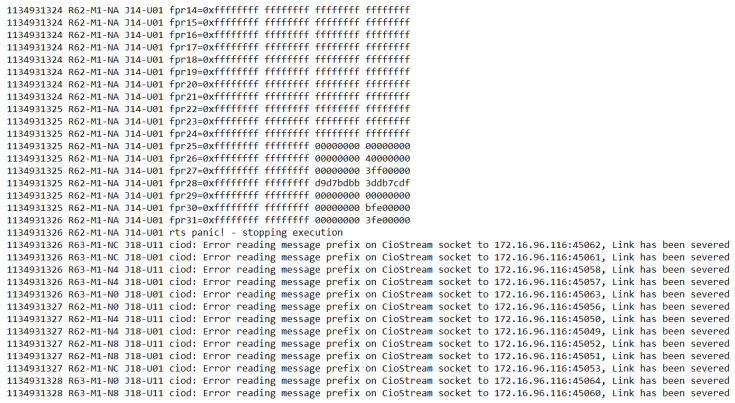


Figura 5.33: **Tupla 363.**

Osservando nel dettaglio il messaggio contenuto nelle entries non è possibile individuare alcuna relazione tra di esse in quanto appartengono a due categorie differenti ed inoltre sono registrate da due nodi separati. Da questa analisi è ragionevole affermare che si tratta di una collision in quanto non esiste alcuna correlazione tra gli eventi registrati nelle due entries.

2) Tupla 298:

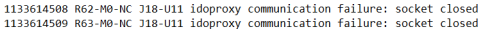


Figura 5.34: **Tupla 298.**

Osservando nel dettaglio il messaggio contenuto nelle entries è possibile individuare relazioni tra di esse in quanto appartengono alla stessa categoria,ma sono registrate da due nodi separati. E' possibile pensare allora, che il fallimento di uno dei due componenti, ha comportato il fallimento successivo: non possiamo

affermare, che questo è un caso di collisione, in quanto è possibile trovare una correlazione tra i due.

5.4.4 Data analysis

L'ultimo passo del processo di FFDA consiste nell'analisi statistica dei dati per poterne estrapolare 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 uno script in Matlab, fornitoci nelle slides presenti nel materiale didattico, che ci ha fornito i seguenti risultati:

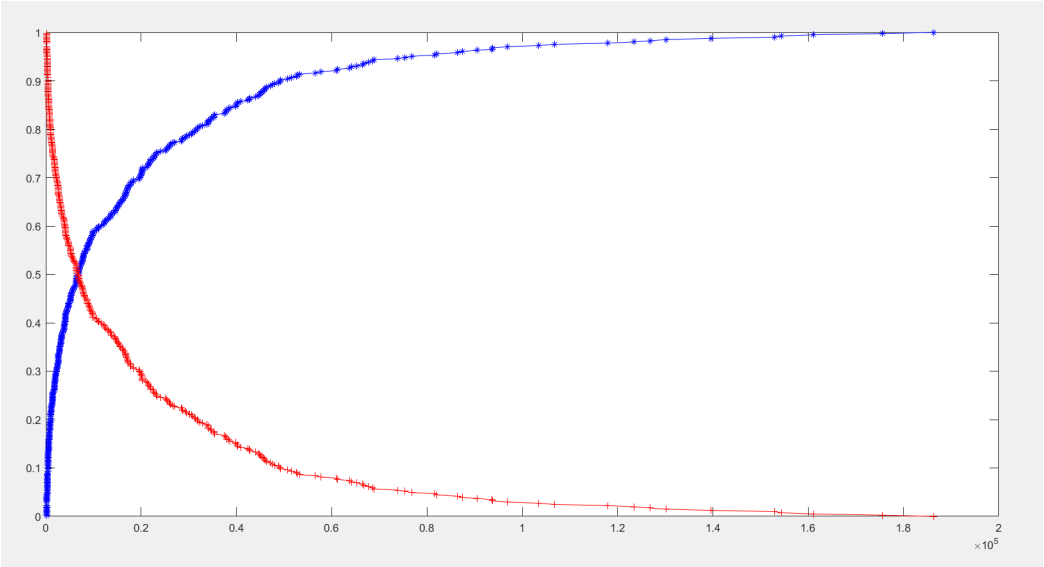


Figura 5.35: **Grafico Reliability - BlueGene/L.**

Al fine di verificare quale delle distribuzioni note è in grado di approssimare al meglio le funzioni di probabilità ottenute, si possono utilizzare due tool diversi di Matlab: Curve Fitting, Distribution Fitting. Il Distibution Fitting da il fitting della distribuzione, il Curve Fitting invece da un modello statistico.
Per quanto riguarda l'utilizzo del Curve Fitting, vediamo:

1) Esponenziale ad 1 parametro:

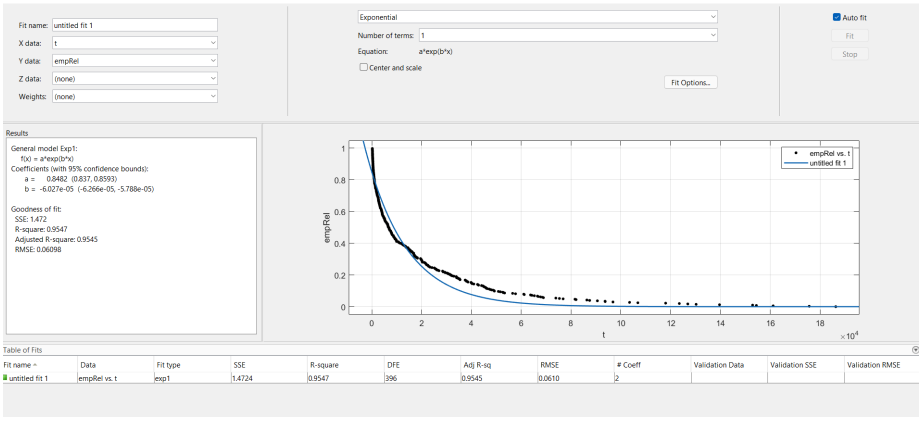


Figura 5.36: **Esponenziale ad un parametro - BlueGene/L.**

2) Esponenziale ad 2 parametri:

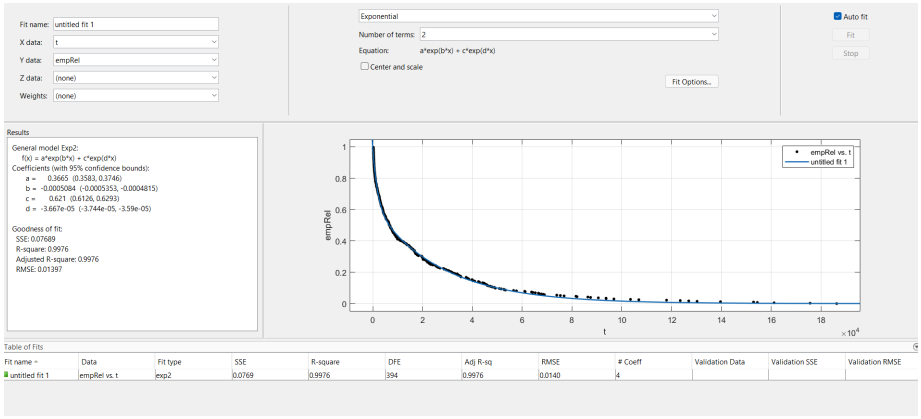


Figura 5.37: **Esponenziale a due parametri - BlueGene/L.**

Confrontando i due tipi di fitting, vediamo come l'esponenziale a due parametri rappresenta molto meglio l'andamento che avevamo ottenuto (questo lo si può notare anche dall'SSE che nella prima figura è più elevato di 1, al contrario, nel secondo caso è quasi pari a 0).
Poichè l'andamento lo rappresentiamo in modo esponenziale, si deduce che c'è stato il fallimento di un componente hardware.
Per verificare che effettivamente i dati ricavati empiricamente seguissero la di-

stribuzione esponenziale, si è utilizzato il test di Kolmogorov-Smirnov.

```
>> [H,P,K]=kstest2(empRel,fittedmodel(t))

H =

  logical

   0


P =

    0.8016


K =

    0.0452
```

Figura 5.38: **K-S test - BlueGene/L.**

Dalla figura 5.38 vediamo come l'ipotesi che il nostro andamento è ben rappresentato dall'esponenziale a due parametri viene accettata (in quanto H=0), con un p-value molto elevato.

Abbiamo voluto vedere se avessimo avuto con l'utilizzo del tool Distribution Fitting, gli stessi risultati.

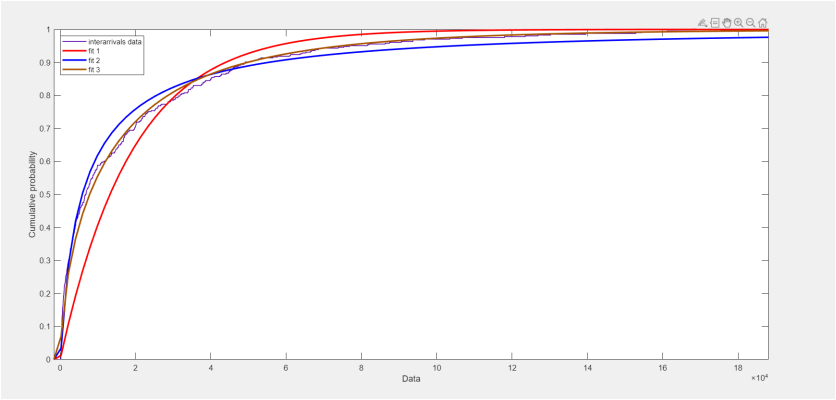


Figura 5.39: **Distribution Fitting - BlueGene/L.**

Nell'immagine vediamo:
- interarrivals (l'andamento degli arrivi);
- fit 1 (andamento Esponenziale);
- fit 2 (andamento Lognormale);
- fit 3 (andamento Weibull).

Probabilmente, l'andamento Weibull (marrone) è quello che si avvicina maggiormente al nostro: dobbiamo fare attenzione, perchè attraverso l'utilizzo del tool Distribution Fitting, stiamo dicendo che effettuando il fitting dei dati ci avviciniamo al grafico di una Weibull, ma non stiamo predicendo nessun andamento futuro, ovvero non stiamo rappresentando un modello.

5.5 Confronto tra supercalcolatori

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: per tale ragione, ci viene da considerare che il sistema BlueGene/L sia più reliable del sistema Mercury. Tale intuizione è confermata dal confronto diretto della reliability dei due sistemi che riportiamo nel grafico in figura 5.40.

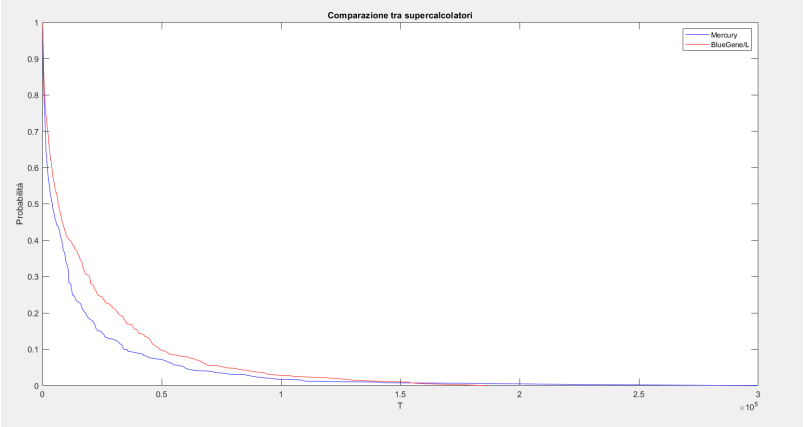


Figura 5.40: **Confronto reliability BlueGene/L e Mercury.**