

Tesina di Impianti di Elaborazione
Prof. Domenico Cotroneo

Luigi Nuzzo - Mat. M63/1059 Giuseppe Porcaro - Mat. M63/1076
Antimo Barbato - Mat. M63/1079

Indice

1	Benchmark	1
1.1	Introduzione	1
1.2	Configurazione	1
1.3	Raccolta dati	1
1.4	Dimensione Campionaria	2
1.5	Distribuzione dei campioni	3
1.6	Differenza statistica	4
1.6.1	50000 corpi	5
1.6.2	200000 corpi	5
1.6.3	500000 corpi	5
1.6.4	1000000 corpi	6
1.6.5	Considerazioni	6
1.7	Confronto	6
2	Workload Characterization	7
2.1	Introduzione	7
2.2	Traccia	7
2.3	Soluzione	8
2.3.1	Filtering	9
2.3.2	Analisi degli outliers	11
2.3.2.1	Analisi di MemFree, Buffer, Cached e Committed_AS	11
2.3.2.2	avgThroughput e VmRSS	11
2.3.2.3	Slab e Active	12
2.3.2.4	Conclusioni	12
2.3.3	Principal Component Analysis	12
2.3.4	Clustering	14
2.3.4.1	PCA & Clustering con 5 componenti	14
2.3.4.2	PCA & Clustering 6 componenti	17
2.3.4.3	PCA & Clustering 7 componenti	19
2.3.5	Conclusioni	21
2.3.5.1	Workload sintetico	22
3	Web Server	23
3.1	Introduzione	23
3.2	Ambiente di lavoro	23
3.3	Capacity Test	24

3.3.1	Parametri Test	24
3.3.2	Risultati	24
3.4	Workload Characterization	25
3.4.1	Fase 1	27
3.4.1.1	Caratterizzazione dei parametri di alto livello	27
3.4.1.2	Caratterizzazione dei parametri di basso livello	30
3.4.2	Fase 2	33
3.4.2.1	Caratterizzazione dei parametri di basso livello	33
3.4.3	Fase 3	33
3.5	Design of Experiments	35
3.5.1	Valutazione del modello	36
3.5.2	Importanza dei fattori	37
3.5.3	Significatività dei fattori	37
3.5.3.1	Normalità	37
3.5.3.2	Omoschedasticità	38
3.5.3.3	Test ANOVA	39
4	Dependability	41
4.1	Esercizio 1	41
4.1.1	Traccia	41
4.1.2	Soluzione	41
4.1.2.1	Teorema dell'Upper Bound	42
4.1.2.2	Conditioning	42
4.1.2.3	MTTF	43
4.2	Esercizio 2	44
4.2.1	Traccia	44
4.2.2	Soluzione	44
4.2.2.1	Confronto	45
4.2.2.2	Equilibrio dei due sistemi	47
4.3	Esercizio 3	48
4.3.1	Traccia	48
4.3.2	Soluzione	49
4.3.2.1	Fault Tree	50
4.4	Esercizio 4	51
4.4.1	Traccia	51
4.4.2	Soluzione Punto 1	51
4.4.3	Soluzione Punto 2	52
4.4.4	Soluzione Punto 3	53
4.4.5	Soluzione Punto 4	54
4.5	Esercizio 5	56
4.5.1	Traccia	56
4.5.2	Soluzione	57
4.5.2.1	Punto 1	57
4.5.2.2	Punto 2	57
4.5.2.3	Punto 3	58
4.5.2.4	Punto 4	60

5 FFDA	61
5.1 Traccia	61
5.2 Field Failure Data Analysis (FFDA)	61
5.3 Mercury	62
5.4 Blue Gene/L	63
5.5 Data Manipulation: Mercury	64
5.5.1 Sensitivity Analysis: Mercury	65
5.6 Data Analysis: Mercury	69
5.6.1 Studio della Reliability	69
5.6.2 Goodness of Fitting	71
5.6.2.1 Test di Kolmogorov-Smirnov	71
5.7 Data Manipulation - Blue Gene	74
5.7.1 Sensitivity Analysis - Blue Gene	74
5.8 Data Analysis: Blue Gene	77
5.8.1 Studio della Reliability	78
5.8.2 Goodness of Fitting	79
5.8.2.1 Test di Kolmogorov-Smirnov	80
5.9 Confronto Mercury e BlueGene	81
5.10 Considerazioni Finali	82
5.10.1 Può la stessa finestra di coalescenza essere usata per differenti nodi (Mercury, BG/L) e differenti errori di categoria (Mercury)?	82
5.10.1.1 Coalescence Window Mercury	85
5.10.1.2 Coalescence Window BlueGene	89
5.10.2 Qual è la relazione tra la reliability del sistema e dei nodi?	91
5.10.2.1 Mercury	91
5.10.2.2 BlueGene	93
5.10.3 Possono esistere dei dependability-bottlenecks?	95
5.10.3.1 Mercury	95
5.10.3.2 BlueGene	96
5.10.4 Nodi funzionali simili hanno parametri di reliability simili?	97
5.10.5 Quali relazioni esistono tra i tipi di errore e i nodi nel caso di Mercury?	98
6 Regression	101
6.1 Introduzione	101
6.2 Dataset EXP1	101
6.2.1 Modello regressivo lineare semplice	101
6.2.2 Verifica delle assunzioni	102
6.2.3 Stima dei parametri regressivi	103
6.3 Dataset EXP2	103
6.3.1 Modello regressivo lineare semplice	104
6.3.2 Verifica delle assunzioni	104
6.3.3 Stima dei parametri regressivi	105
6.4 Dataset os1	106
6.4.1 Modello regressivo lineare semplice	106
6.4.2 Verifica delle assunzioni	108
6.4.3 Stima dei parametri regressivi	109

6.5	Dataset os2	110
6.5.1	Modello regressivo lineare semplice	110
6.5.2	Verifica delle assunzioni	112
6.5.3	Stima dei parametri regressivi	113
6.6	Dataset os3	114
6.6.1	Modello regressivo lineare semplice	114
6.6.2	Verifica delle assunzioni	116
6.6.3	Stima dei parametri regressivi	117
6.7	Dataset VMres1	118
6.7.1	Modello regressivo lineare semplice	118
6.7.2	Verifica delle assunzioni	118
6.7.3	Stima dei parametri regressivi	119
6.7.4	Failure prediction	119
6.8	Dataset VMres2	120
6.8.1	Modello regressivo lineare semplice	120
6.8.2	Verifica delle assunzioni	120
6.8.3	Stima dei parametri regressivi	121
6.8.4	Failure prediction	121
6.9	Dataset VMres3	122
6.9.1	Modello regressivo lineare semplice	122
6.9.2	Verifica delle assunzioni	122
6.9.3	Stima dei parametri regressivi	123
6.9.4	Failure prediction	123

Capitolo 1

Benchmark

1.1 Introduzione

Si vuole effettuare il confronto tra due sistemi di elaborazione differenti utilizzando il **Benchmark nbody**. Esso è utilizzato per valutare le prestazioni di un sistema di elaborazione poiché richiede una grande quantità di calcoli per modellare il moto di un prefissato numero di corpi celesti in un sistema gravitazionale. L'obiettivo dell'homework è quello di dimostrare che i tempi di esecuzione valutati durante l'esecuzione indipendente di nbody risultino **statisticamente differenti** in modo da effettuare il confronto tra i due sistemi di elaborazione.

1.2 Configurazione

Di seguito vengono elencate le specifiche delle due macchine su cui è stato eseguito il benchmark.

	Processore	Core	Thread	Ram	Sistema Operativo
M1	Intel® Core™ i5-5200U CPU @ 2.2GHz	2	4	4 GB	Windows 10 22H2
M2	Intel® Core™ i7-3537U CPU @ 2.5GHz	2	4	12 GB	Windows 10 22H2

Tabella 1.1: Configurazione macchine da testare

1.3 Raccolta dati

La prima condizione fondamentale da garantire è l'indipendenza dei campioni raccolti in modo da applicare il Teorema del Limite Centrale.

Le macchine sono state riavviate dopo ogni esecuzione per poter garantire l'indipendenza dei dati. Nello specifico, il comando utilizzato per le misurazioni è il seguente:

```
./launch_nbody.sh -r 5 -n *
```

con:

- **r**: numero di ripetizioni;
- **n**: numero di corpi da simulare.

Nel dettaglio, sono state raccolte **30 misurazioni indipendenti** da 5 ripetizioni ciascuna per **50000, 200000, 500000, 1000000** corpi (50 mila, 200 mila, 500 mila, 1 milione).

1.4 Dimensione Campionaria

Le osservazioni raccolte costituiscono il **precampione** che viene utilizzato per calcolare la **dimensione campionaria**. Ciò ci permette di avere certezza di ricadere nell'ipotesi in cui i campioni prelevati esprimano un'accuratezza dello 0.05% con un intervallo di confidenza del 95%. La formula è la seguente:

$$n = \left(\frac{100 * z_{\frac{\alpha}{2}} * s}{r * \bar{x}} \right)^2$$

con:

- $z_{\frac{\alpha}{2}}$ che è uguale a 1.96 ottenuto tramite la tabella di riferimento “z-table”;
- s che è la deviazione standard del pre-campione;
- r che è l'accuratezza desiderata;
- \bar{x} che è la media del pre-campione.

Si visualizzano i risultati nelle tabelle successive:

MACCHINA 1			
Numero corpi	Media	Deviazione standard	Dimensione campionaria
50000	18775.05	469.41	0.96
200000	71484.44	623.96	0.11
500000	177413.65	1816.43	0.16
1000000	353358.13	2907.86	0.10

Tabella 1.2: Dimensione campionaria MACCHINA 1

MACCHINA 2			
Numero corpi	Media	Deviazione standard	Dimensione campionaria
50000	22498.56	482.72	0.70
200000	90302.60	1310.44	0.32
500000	235688.4	1348.23	0.05
1000000	473553.02	2865.69	0.05

Tabella 1.3: Dimensione campionaria MACCHINA 2

Come si può vedere, il valore maggiore corrisponde all'esperimento fatto sulla macchina 1 per 50000 corpi. Tuttavia il numero risulta molto più piccolo del numero delle osservazioni del pre-campione quindi si è già nelle condizioni richieste e si procede senza dover effettuare nuovamente il campionamento.

1.5 Distribuzione dei campioni

Utilizzando il software JMP, si valuta inizialmente la normalità dei campioni e successivamente la loro differenza statistica. Per la valutazione della normalità sono stati eseguiti dei test visivi sui Q-Q plot. Possiamo visualizzare i grafici nelle figure successive.

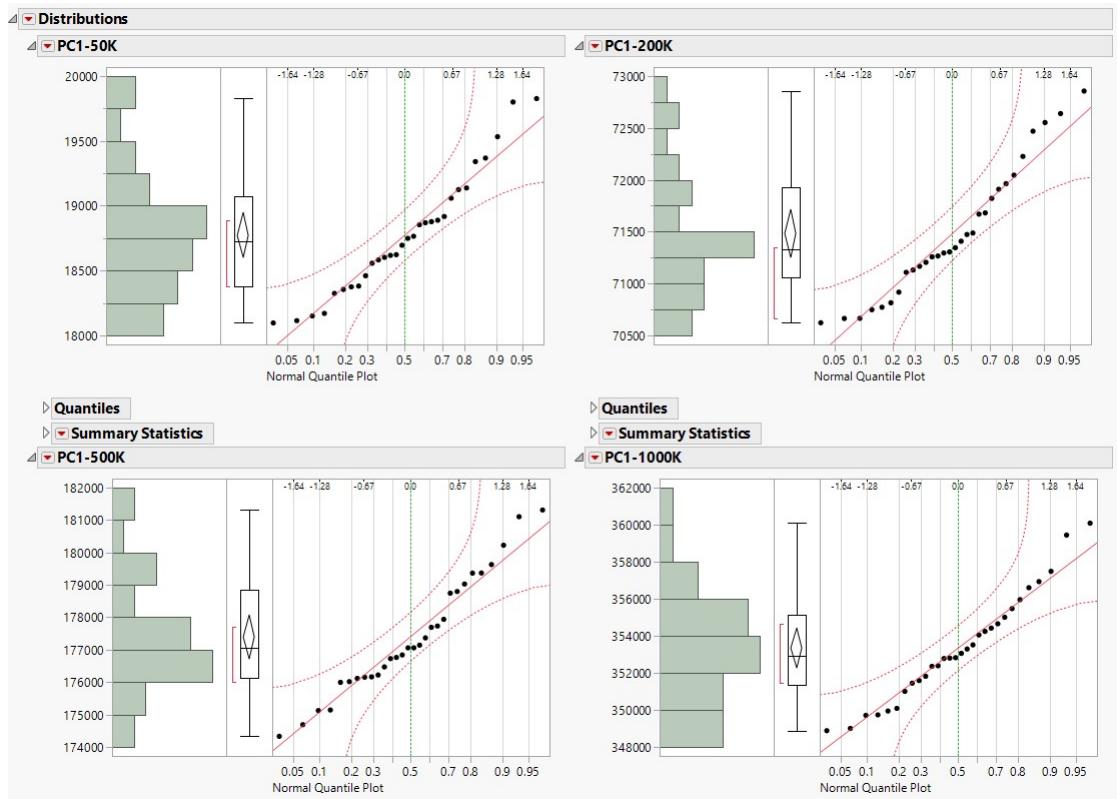


Figura 1.1: Q-Q plot MACCHINA 1

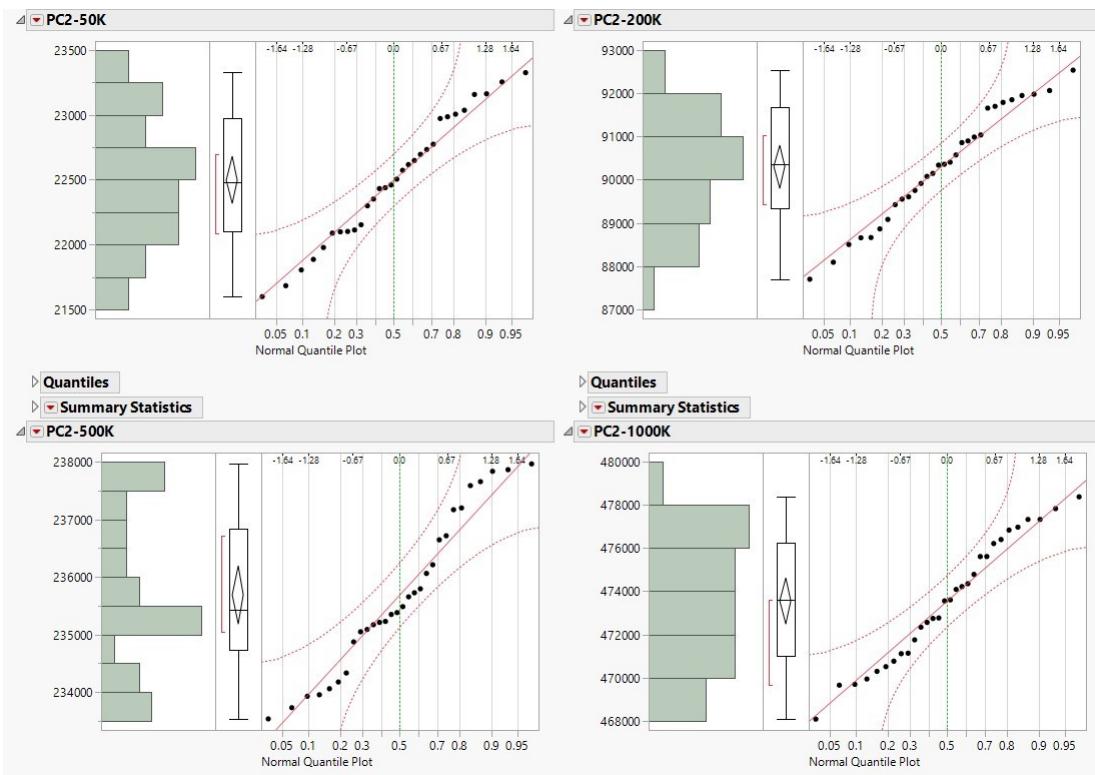


Figura 1.2: Q-Q plot MACCHINA 2

Come si può notare dai diagrammi, in tutti gli esperimenti, le osservazioni dei campioni rientrano nelle bande di confidenza della distribuzione normale. Quindi, si può dire che le distribuzioni possono essere considerate normali, le osservazioni dei campioni indipendenti e identicamente distribuite tra loro. Riassumendo:

	MACCHINA 1	MACCHINA 2
Numero di corpi	Normalità	Normalità
50000	Non rigettata	Non rigettata
200000	Non rigettata	Non rigettata
500000	Non rigettata	Non rigettata
1000000	Non rigettata	Non rigettata

Tabella 1.4: Tabella normalità

1.6 Differenza statistica

Per poter effettuare un confronto, bisogna verificare la differenza statistica tra le due distribuzioni. Accertata la normalità, si può utilizzare un test parametrico, nel caso in esame si utilizzerà un **paired t-test** poiché si sta effettuando un confronto 1:1. L'ipotesi nulla del t-test prevede che la distribuzione della differenza dei dati nei campioni considerati sia una normale con media diversa da zero. Se l'ipotesi non è rigettata, i campioni non possono essere statisticamente differenti e quindi non si può assumere che provengano da popolazioni differenti.

Sempre tramite il software JMP si va a testare la differenza delle distribuzioni per ogni dimensione considerata.

1.6.1 50000 corpi

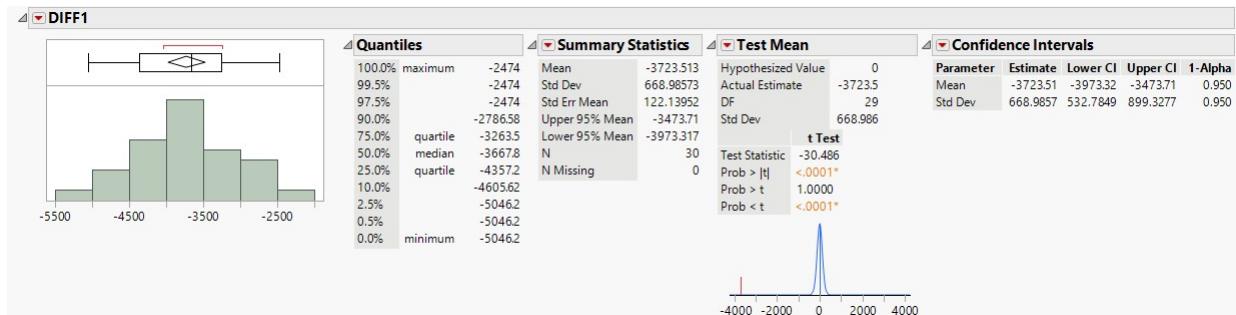


Figura 1.3: Test JMP 50000

1.6.2 200000 corpi

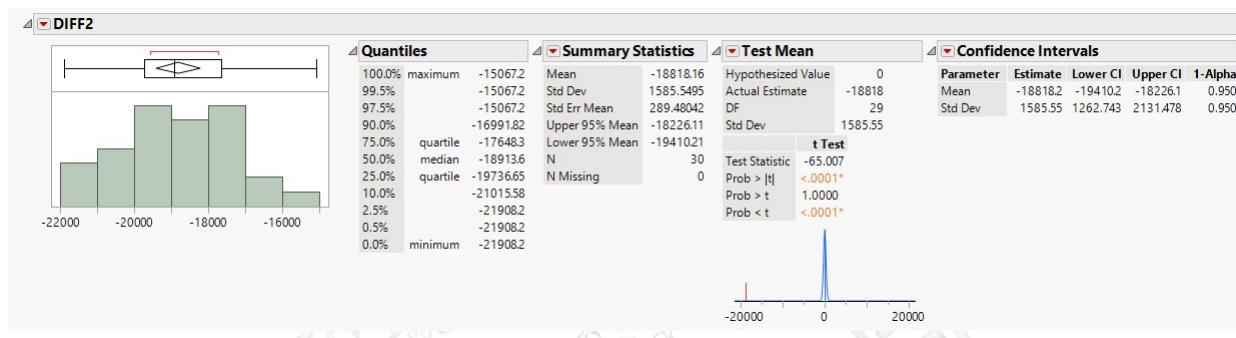


Figura 1.4: Test JMP 200000

1.6.3 500000 corpi

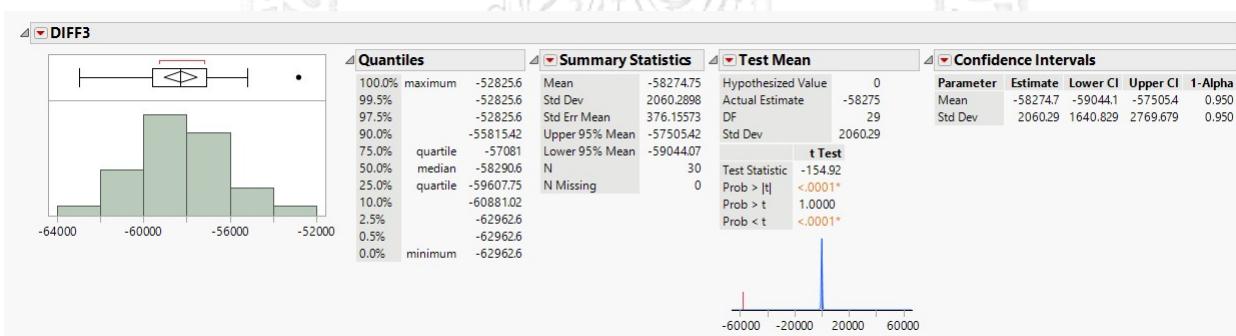


Figura 1.5: Test JMP 500000

1.6.4 1000000 corpi

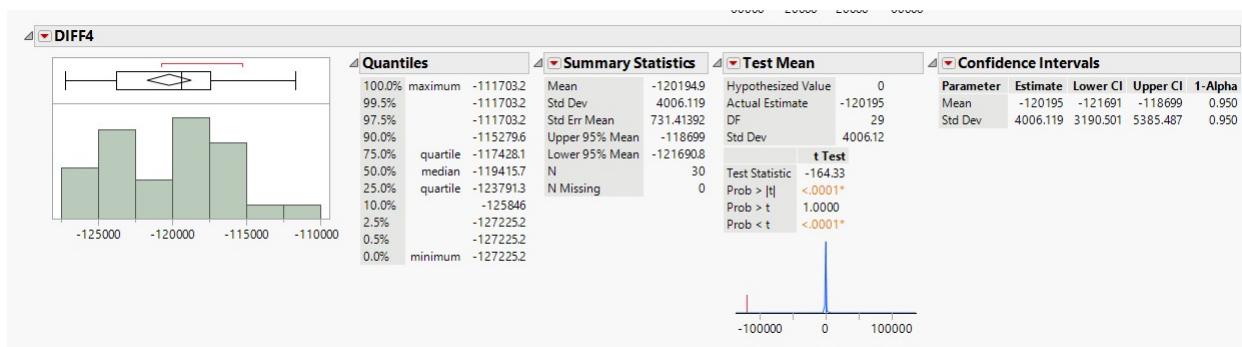


Figura 1.6: Test JMP 1000000

1.6.5 Considerazioni

Come si può vedere dal test, l'ipotesi nulla è rigettata per tutte le dimensioni e i vari intervalli di confidenza non contengono lo zero. Dunque si può concludere che i campioni provengono da popolazioni differenti e quindi possono essere confrontati.

1.7 Confronto

Siccome i campioni seguono una distribuzione normale, il confronto tra gli stessi esperimenti può essere effettuato rispetto alla media. I risultati si possono vedere nella successiva tabella:

Numero di corpi	MACCHINA 1	MACCHINA 2	Confronto
	Media	Media	
50000	18775.05	22498.56	M1 < M2
200000	71484.44	90302.60	M1 < M2
500000	177413.65	235688.4	M1 < M2
1000000	353358.13	473553.02	M1 < M2

Tabella 1.5: Tabella confronto

Si può infine concludere che il sistema 1 con processore i5-5200U risulta presentare tempi minori per le operazioni di benchmark nbody rispetto al sistema 2 con processore i7-3537U. Questa conclusione probabilmente è dovuta alla differenza generazionale dei singoli processori.

Capitolo 2

Workload Characterization

2.1 Introduzione

L’obiettivo della Workload Characterization è quello della caratterizzazione di un Workload reale, ossia la creazione di un Workload Sintetico (a partire da quello reale), con lo scopo di conservare quanta più varianza possibile, nonostante la riduzione dei dati.

2.2 Traccia

Dato un workload reale, trovarne uno sintetico rispettando i seguenti requisiti:

- Trovare il miglior trade-off tra numero di PCA e numero di Cluster;
- Analisi di sensitività, pensare a varie alternative per calcolare facilmente la devianza persa dopo PCA e Clustering.

Il dataset fornito è composto da 24 colonne e 5000 righe. Ogni colonna corrisponde a un parametro che è stato ottenuto interrogando il file system proc. Si tratta di un filesystem virtuale gestito dal kernel di Linux a scopo di monitoraggio e al suo interno possiamo trovare alcuni file che descrivono lo stato corrente del sistema e altri associati ai processi in esecuzione:

- proc/pid/status contiene informazioni sulle risorse utilizzate da uno specifico processo:
 - VmPeak: dimensione di picco della memoria virtuale;
 - VmSize: dimensione totale del programma;
 - VmHWM: limite massimo di memoria utilizzato;
 - VmRSS: utilizzo della memoria corrente;
 - VmPTE: dimensione delle entry della tabella delle pagine;
 - VmSwap: quantità di swap utilizzata da dati privati anonimi;
 - Threads: numero di thread.
- proc/meminfo contiene informazioni sullo stato della memoria.
 - MemFree: quantità di RAM fisica, in kilobyte, lasciata inutilizzata dal sistema;

- Buffers: quantità di RAM fisica in KB utilizzata per i buffer di file;
- Cached: quantità di RAM fisica, in kilobyte, utilizzata come memoria cache;
- SwapCached: quantità di swap, in kilobyte, utilizzata come memoria cache;
- Active: quantità totale di memoria buffer o cache di pagina, in kilobyte, che è in uso attivo. Questa è la memoria che è stata utilizzata di recente e di solito non viene recuperata per altri scopi;
- Inactive: quantità totale di memoria buffer o cache di pagina, in kilobyte, che è libera e disponibile. Questa è la memoria che non è stata utilizzata di recente e può essere recuperata per altri scopi;
- Dirty: quantità totale di memoria, in kilobyte, in attesa di essere riscritta sul disco;
- Writeback: quantità totale di memoria, in kilobyte, che viene riscritta attivamente sul disco
- AnonPages: quantità totale di memoria, in kilobyte, utilizzata dalle pagine che non sono supportate da file e sono mappate nelle tabelle delle pagine dello spazio utente;
- Mapped: quantità totale di memoria, in kilobyte, che è stata utilizzata per mappare dispositivi, file o librerie utilizzando il comando mmap. Questa funzione crea una nuova corrispondenza nello spazio degli indirizzi virtuali del processo chiamante;
- Slab: quantità totale di memoria, in kilobyte, utilizzata dal kernel per memorizzare nella cache le strutture dati per uso personale (del kernel);
- PageTables: quantità totale di memoria, in kilobyte, dedicata al livello della tabella delle pagine più basso;
- Committed_AS: quantità totale di memoria, in kilobyte, stimata per completare il carico di lavoro. Questo valore rappresenta il valore dello scenario peggiore e include anche la memoria swap;
- KernelStack: memoria consumata dagli stack del kernel di tutti i task;
- Shmem: memoria totale utilizzata dalla memoria condivisa (shmem) e da tmpfs (un tipo di memoria temporanea tipica dei sistemi operativi Unix-like);
- MemAvailable: una stima della quantità di memoria disponibile per l'avvio di nuove applicazioni, senza swapping. La stima tiene conto del fatto che il sistema ha bisogno di una certa cache di pagine per funzionare bene e che non tutti gli slab recuperabili saranno recuperati;
- Unevictable: memoria allocata per lo spazio utente che non può essere recuperata, come le backing page di ramfs;
- SwapFree: memoria che è stata espulsa dalla RAM e si trova temporaneamente sul disco.

2.3 Soluzione

Per risolvere questo esercizio, è stato previsto l'utilizzo del tool JMP per l'analisi del dataset al fine di ridurre il numero di dati. Dopo un'analisi preliminare sui dati, la riduzione del dataset è stata effettuata usando due tecniche :

- Principal Component Analysis (PCA);
- Clustering.

2.3.1 Filtering

Il primo passo per il filtraggio dei dati è stato quello di caricare gli stessi su JMP per poterne effettuare un analisi tabellare, il controllo delle colonne ha poi rivelato la presenza di alcune con varianza nulla, poiché costanti. In particolare, osservando anche la figura seguente sono state eliminate dall'analisi le colonne relative a: **AnonPages**, **avgLatency** ed **Errors**;

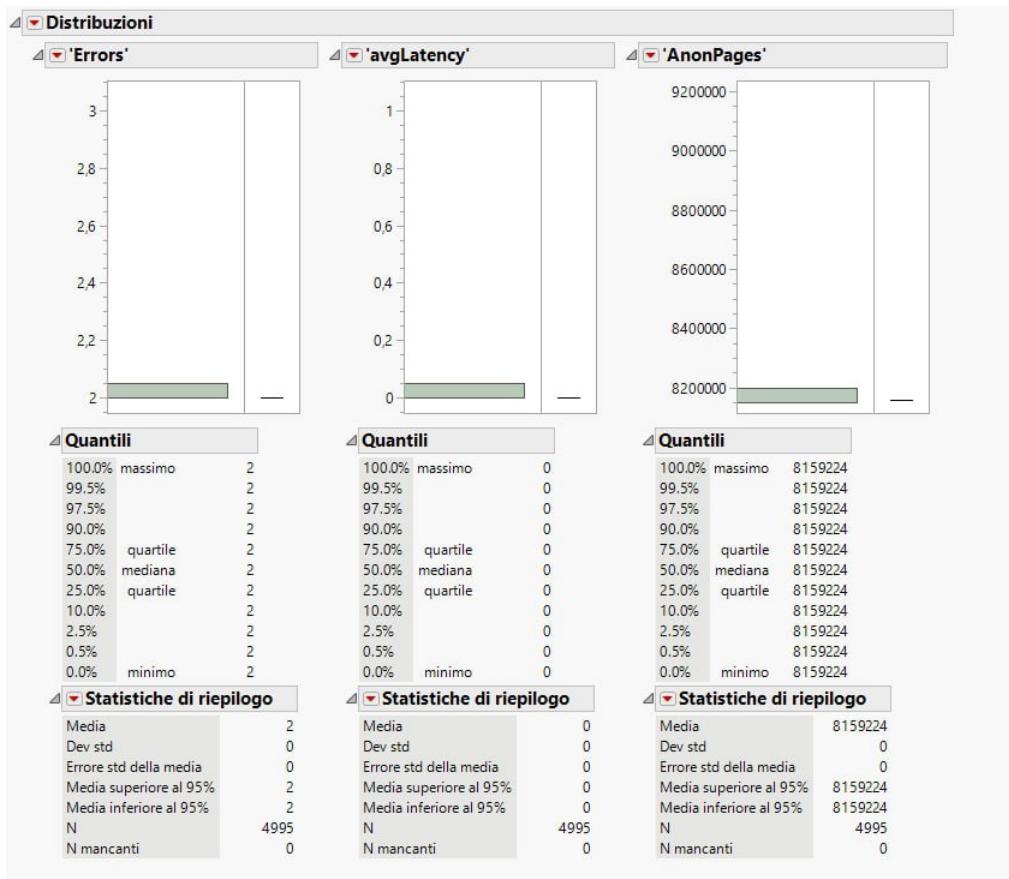


Figura 2.1: Distribuzioni con varianza nulla

Inoltre, si è notato che gli attributi **MemFree** e **WriteBack** presentano praticamente gli stessi valori per ogni osservazione e di conseguenza la stessa distribuzione, come è possibile visualizzare in figura:

CAPITOLO 2. WORKLOAD CHARACTERIZATION

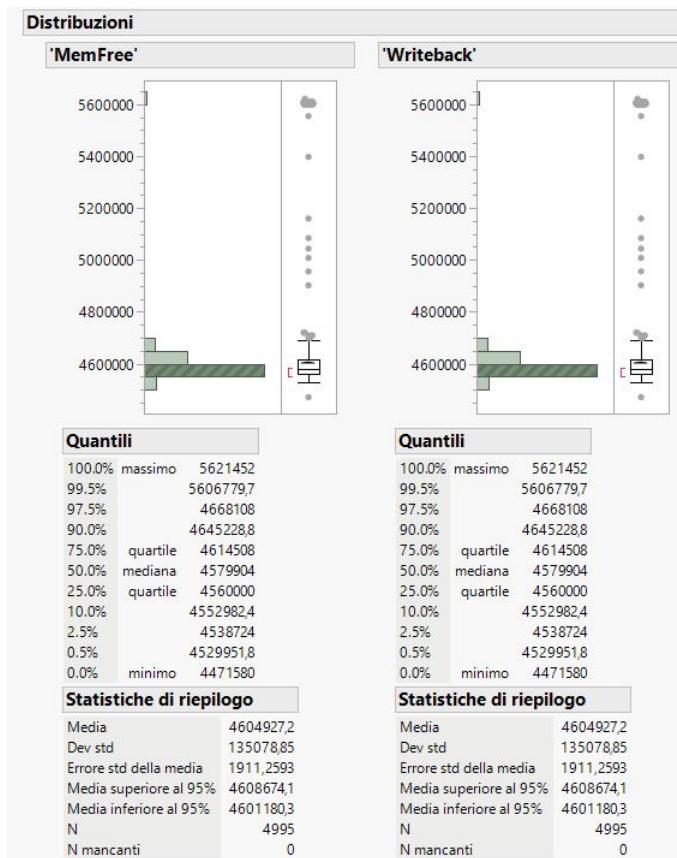


Figura 2.2: Distribuzioni MemFree e WriteBack

Per verificare la correttezza dell'affermazione è stata generata la matrice di correlazione tra i parametri; come si può vedere, i valori in verde indicano l'uguaglianza dei due considerati in precedenza.

Attributi	VmPeak'	VmSize'	VmHWM'	VmRSS'	VmPTE'	Threads'	MemFree'	Buffers'	Cached'	Active'	Inactive'	Dirty'	Writeback'
VmPeak'	1	0.9899537	0.98554409	0.9389232	0.9910236	0.6311729	-0.46678019	0.6530157	0.3068698	-0.00123	0.7395664	0.250779	-0.466780187
VmSize'	0.98995367	1	0.97992376	0.9627658	0.9906082	0.6143379	-0.4892154	0.6376366	0.3385964	-0.00207	0.7455846	0.281836	-0.489215398
VmHWM'	0.98554409	0.9799238	1	0.9675714	0.9773329	0.5954915	-0.38717785	0.6102408	0.2227246	-0.00245	0.6770926	0.169313	-0.387177846
VmRSS'	0.93892319	0.9627658	0.96757139	1	0.9416418	0.56060417	-0.37325564	0.5484077	0.2250454	-0.00388	0.6399967	0.172082	-0.373255635
VmPTE'	0.99102362	0.9906082	0.97733287	0.9416418	1	0.6323648	-0.41453049	0.6178751	0.2541429	-0.00143	0.6995265	0.195755	-0.41453049
Threads'	0.63117287	0.6143379	0.59549154	0.5660417	0.6323648	1	-0.22542075	0.4706622	0.0846528	0.012528	0.4577064	0.051224	-0.225420748
MemFree'	-0.4667802	-0.489215	-0.38717785	-0.373256	-0.41453	-0.2254207	1	-0.724307	-0.968062	-0.00075	-0.9064017	-0.96453	1
Buffers'	0.65301571	0.6376366	0.61024063	0.5484077	0.6178751	0.4706622	-0.72430694	1	0.532009	-0.01161	0.8926784	0.555448	-0.724306944
Cached'	0.30686975	0.3385964	0.22272463	0.2250454	0.2541429	0.0846528	-0.96806221	0.532009	1	0.004649	0.7825302	0.991972	-0.968062211
Active'	-0.0012276	-0.002066	-0.00245448	-0.003884	-0.001426	0.0125282	-0.0074988	-0.011605	0.0046494	1	-0.0055811	0.004173	-0.000749864
Inactive'	0.73956643	0.7455846	0.6770926	0.6399967	0.6995265	0.4577064	-0.90640172	0.8926784	0.7825302	-0.00558	1	0.763093	0.906401716
Dirty'	0.2507787	0.2818357	0.16931268	0.1720817	0.1957552	0.0512238	-0.96452938	0.5554478	0.9919723	0.004173	0.7630934	1	-0.964529384
Writeback'	-0.4667802	-0.489215	-0.38717785	-0.373256	-0.41453	-0.2254207	1	-0.724307	-0.968062	-0.00075	-0.9064017	-0.96453	1
Mapped'	-0.0604589	-0.052563	-0.05480048	-0.043924	-0.049135	-0.0233997	0.061503711	-0.128963	-0.046824	-0.00193	-0.0285485	-0.08205	0.061503711
Slab'	-0.0112512	-0.008091	-0.01214848	-0.008951	-0.008744	-0.0012734	0.005336588	-0.033881	-0.001913	-0.0005	0.0243948	-0.02432	0.005336588
PageTables'	0.88575281	0.9090642	0.91604298	0.9487819	0.8910028	0.5442344	-0.33484451	0.4969196	0.1907935	-0.00405	0.6131125	0.128045	-0.334844512
Committed_AS'	0.06959449	0.0672123	0.0774691	0.0705216	0.0752894	0.1034499	-0.01727628	0.1064681	-0.03795	-0.00635	0.1456219	-0.06955	-0.017276276
NumOfAllocFH'	0.40545586	0.426144	0.31482961	0.2981193	0.3524949	0.2187698	-0.9847945	0.636412	0.9796509	0.005024	0.8547257	0.970442	-0.984794505
proc-fd'	0.50158704	0.5104596	0.51493226	0.5179689	0.515548	0.3488006	-0.1065727	0.2221122	0.0170749	-0.00316	0.3069002	-0.03439	-0.106572703
avgThroughput'	0.96499339	0.9665955	0.94413632	0.9112683	0.9640431	0.637314	-0.48971334	0.6361764	0.3375782	-0.00038	0.7533081	0.275565	-0.489713339
avgElapsed'	-0.4491222	-0.424969	-0.43663636	-0.390473	-0.44366	-0.356932	0.13543599	-0.298274	-0.046964	0.010464	-0.2893881	-0.02477	0.13543599

Figura 2.3: Matrice di correlazione

2.3.2 Analisi degli outliers

2.3.2.1 Analisi di MemFree, Buffer, Cached e Committed_AS

Inizialmente si è scelto di analizzare per primi gli outliers dei seguenti attributi: **MemFree**, che come abbiamo detto indica la quantità di memoria RAM attualmente disponibile, **Buffer**, che indica l'archiviazione temporanea per blocchi di dischi non elaborati, **Cached**, che indica i file letti nel disco e memorizzati nelle cache e infine **Committed_AS**, che indica l'intera quantità di memoria RAM allocata per i processi.

Come si evince nella successiva figura, sono stati analizzati gli outliers corrispondenti alla prima osservazione, ossia la riga 1. Si può pensare che questa analisi rifletta uno stato di basso carico del sistema dovuto a un recente avvio. Infatti MemFree presenta un valore massimo e gli altri tre il loro valore minimo.

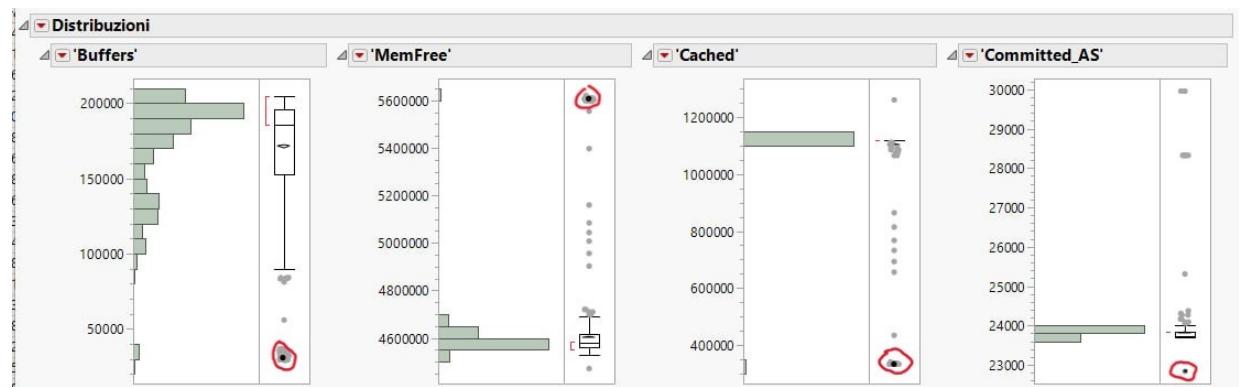


Figura 2.4: Prima analisi outliers primi parametri

Andando ad analizzare la distribuzione dell'attributo MemFree e valutando gli outliers nel piano, si nota che questi corrispondono all'intervallo di osservazioni con indice [2:94] e risultano essere outliers anche per gli altri tre attributi, come possiamo vedere:

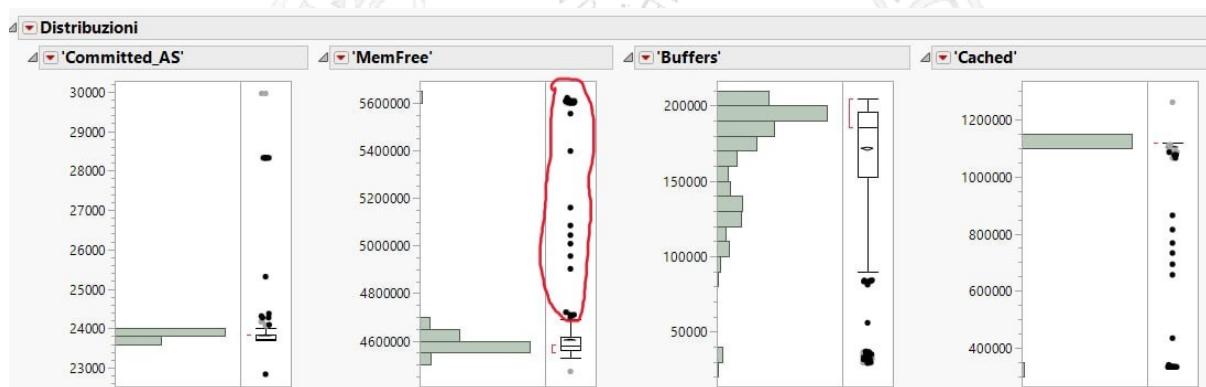


Figura 2.5: Seconda analisi outliers primi parametri

2.3.2.2 avgThroughput e VmRSS

Gli attributi da analizzare adesso sono due: **avgThroughput** che indica il throughput medio del sistema e **VmRSS** che indica la dimensione della memoria RAM effettivamente tenuta in uso dai

processi. Da come si può osservare nella figura successiva, per questi valori il sistema presenta un'elevata memoria in uso e allo stesso tempo i più alti valori per l'attributo avgThroughput. Si può notare che in questa fase il sistema è sottoposto a un carico più oneroso.

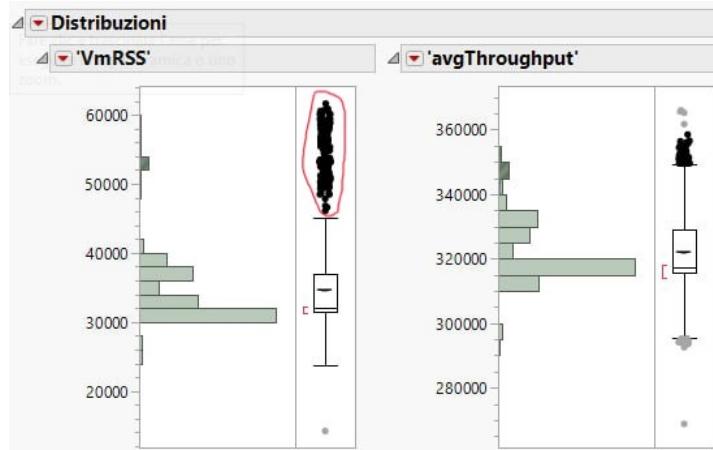


Figura 2.6: Analisi outliers secondi parametri

2.3.2.3 Slab e Active

Infine, si sono analizzati gli outliers di **Slab**, che indica la quantità di cache nel kernel riservata alla struttura dati e **Active** che indica la quantità totale di memoria buffer o cache di pagina, in kilobyte, che è in uso attivo. In questo caso, un outlier si trova nel punto in cui si è identificata la fine della fase di “avvio” (riga 90). Un altro outlier si trova nella fase “centrale” e non risulta essere di particolare interesse poiché risulta appartenere a un’osservazione che presenta dei valori in media nella distribuzione per tutti gli altri gli attributi. Gli ultimi outliers si trovano alla fine ma anche in questo caso non risultano significativi.

Anche per quanto riguarda Active si può applicare lo stesso ragionamento, in quanto i valori non sembrano di particolare rilevanza.

2.3.2.4 Conclusioni

Non conoscendo l’obiettivo dell’analisi del sistema si è scelto di non filtrare nessuno degli outliers discussi in precedenza. Non avendo conoscenza del contesto di studio, il filtraggio anche di un singolo outliers potrebbe ridurre significativamente le informazioni di interesse.

2.3.3 Principal Component Analysis

Le tecniche di **Principal Component Analysis** consentono di ridurre gli attributi di un dataset identificando al suo interno quelle che sono, appunto, le componenti principali. L’idea è quella di individuare nuove componenti incorrelate tra loro attraverso una trasformazione di stato che permetta di ordinarle da quella che esprime più variabilità a quella che esprime meno variabilità. Tale trasformazione è utile perché riusciamo a capire quante e quali sono le componenti tramite cui riduciamo la dimensionalità del problema tenendo traccia della varianza persa. Il tool JMP fornisce i seguenti output che vediamo in figura:

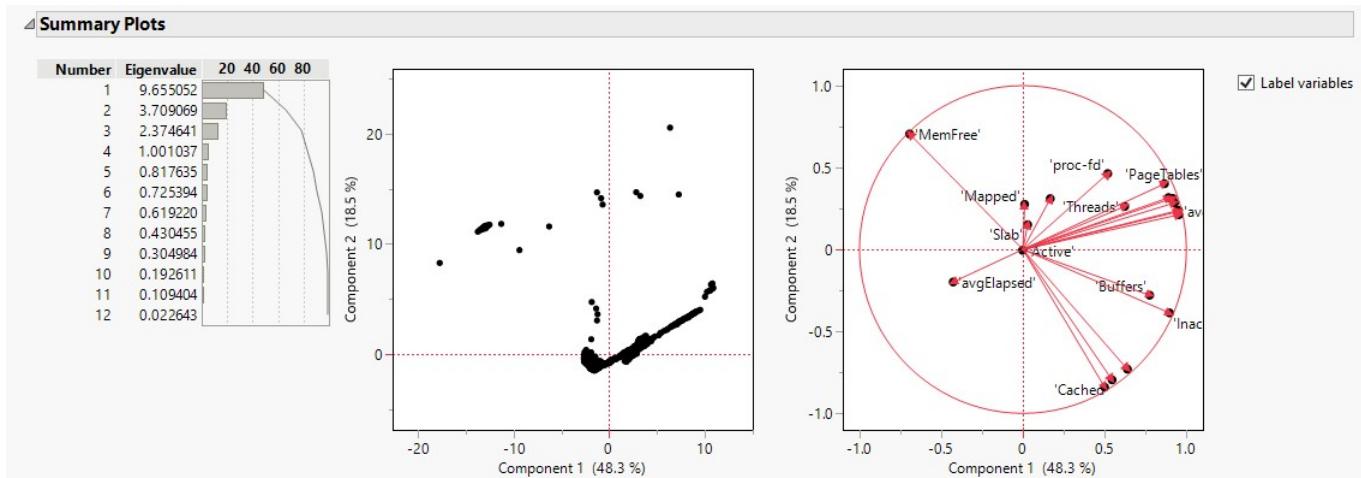


Figura 2.7: Diagrammi di riepilogo PCA

Il primo grafico è il **Diagramma a barre** che mostra la varianza conservata da ogni componente principale; il secondo è il **Loading Plot** che mostra una matrice di rappresentazione bidimensionale dei fattori di carico. Infine il terzo è lo **Score Plot** che mostra una matrice di dispersione per coppie di componenti principali: spiega quindi come si distribuiscono gli elementi in base a due componenti principali selezionate.

Il problema da considerare è la scelta del numero di componenti principali da selezionare, cercando di preservare quanta più varianza possibile. Possiamo visualizzare in due modi la percentuale cumulativa: tramite diagramma a barre generato da JMP e un plot generato da Matlab in cui si può vedere attraverso una curva l'andamento della varianza conservata dopo la PCA.

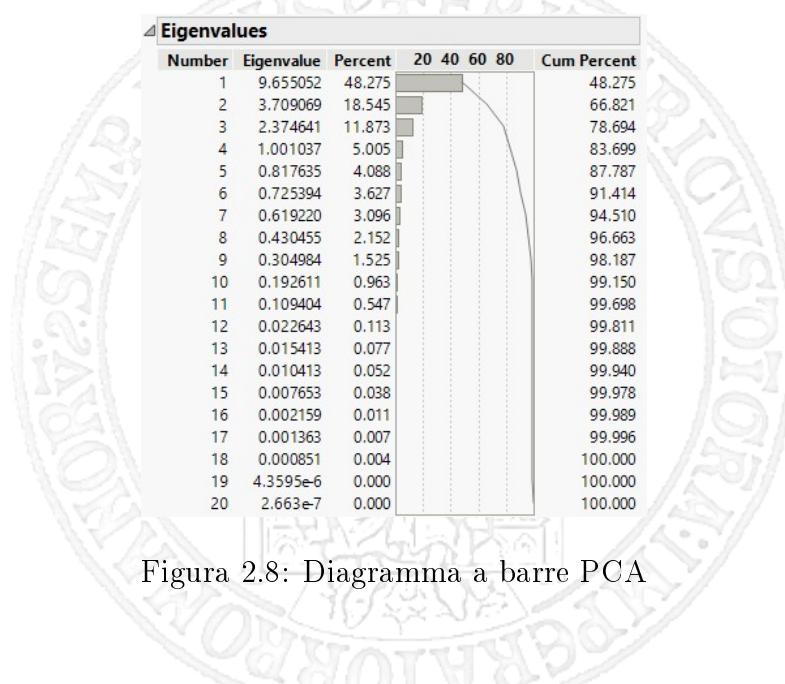


Figura 2.8: Diagramma a barre PCA

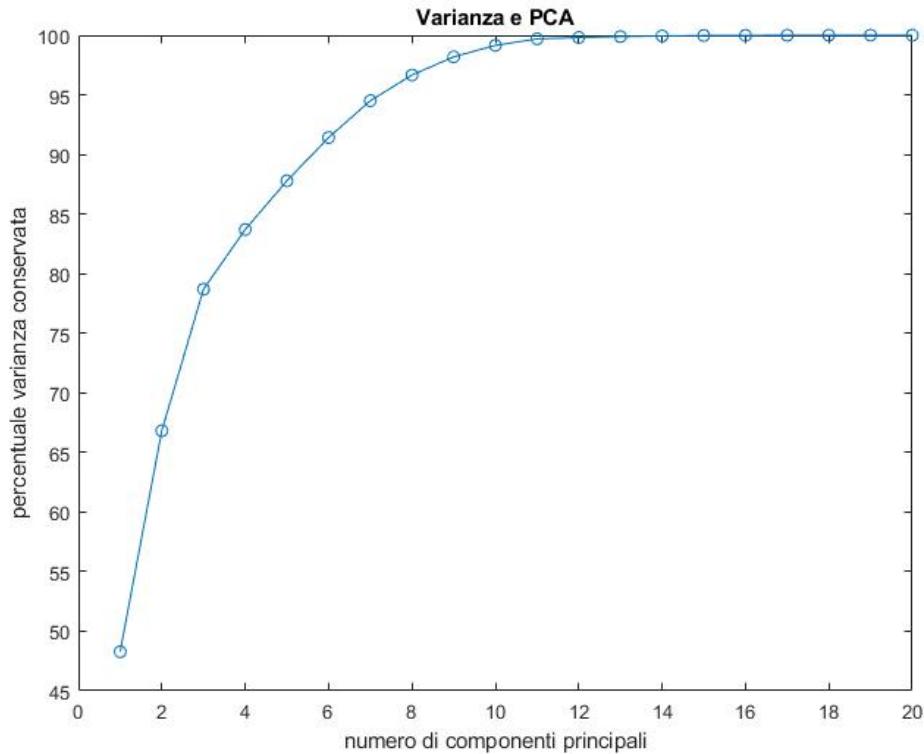


Figura 2.9: Trend varianza e PCA

Dal grafico si può notare che il primo valore utile per le componenti principali è 5 che spiega l'87.7% della varianza. Tuttavia per ricercare il miglior trade-off si è scelto di confrontare i risultati per numero di componenti principali pari a 6 e 7 che spiegano rispettivamente il 91.4% e il 94.5% della varianza.

2.3.4 Clustering

Estratte le componenti principali selezionate si è passati alle operazioni di clustering: nel dettaglio è stato effettuato un clustering gerarchico con il **Metodo di Ward**. Per definizione al termine del processo di clustering si avrà una **varianza intraccluster** minima mentre quella **intracluster** sarà massima.

JMP darà come output un **dendrogramma**, che si vedrà successivamente per i vari esempi.

2.3.4.1 PCA & Clustering con 5 componenti

Nella figura successiva possiamo notare il dendrogramma generato da JMP dopo la scelta di 5 componenti principali, sapendo che dopo l'operazione di PCA la percentuale di varianza conservata è dell'87.7%.

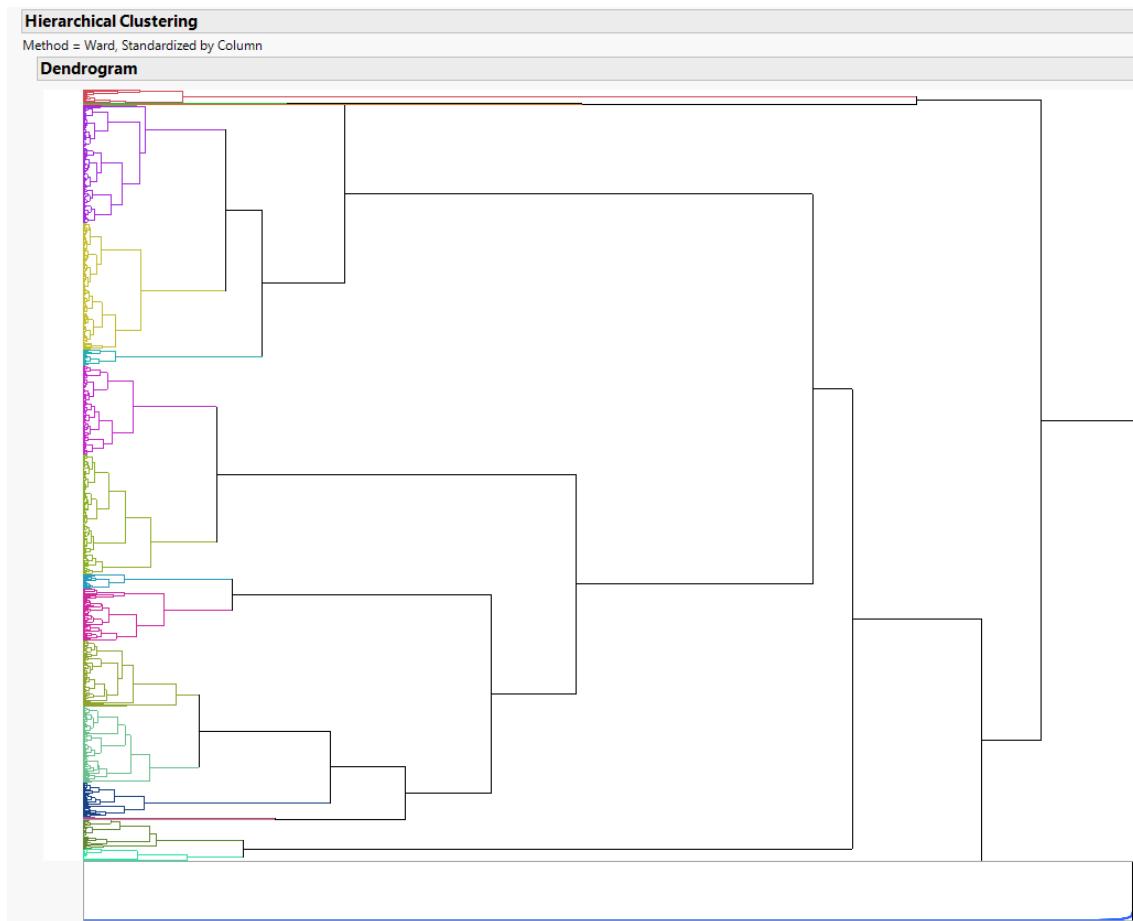
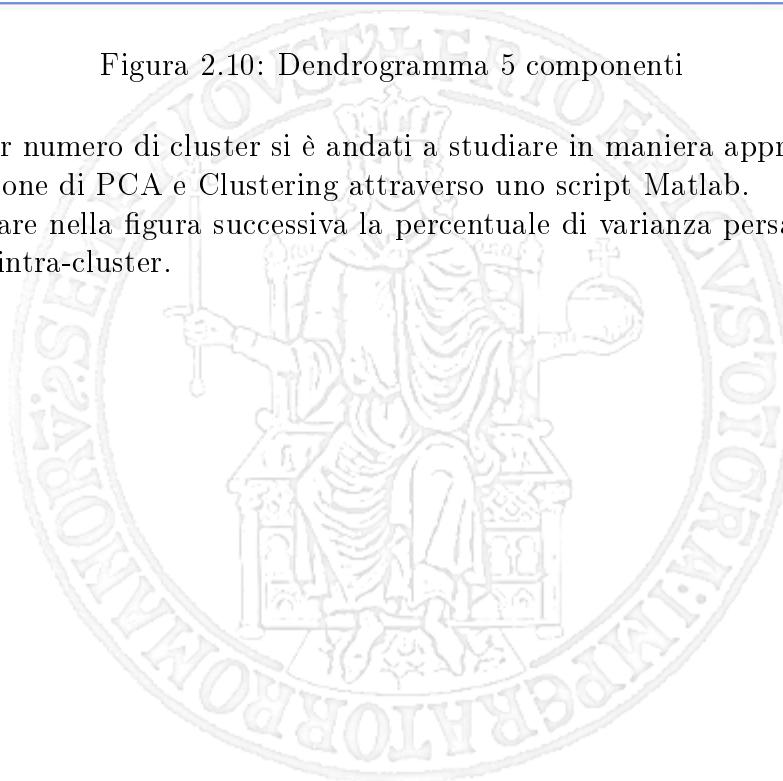


Figura 2.10: Dendrogramma 5 componenti

Per cercare il miglior numero di cluster si è andati a studiare in maniera approfondita la varianza persa dopo l'operazione di PCA e Clustering attraverso uno script Matlab.

Come possiamo notare nella figura successiva la percentuale di varianza persa si può visualizzare tramite la varianza intra-cluster.



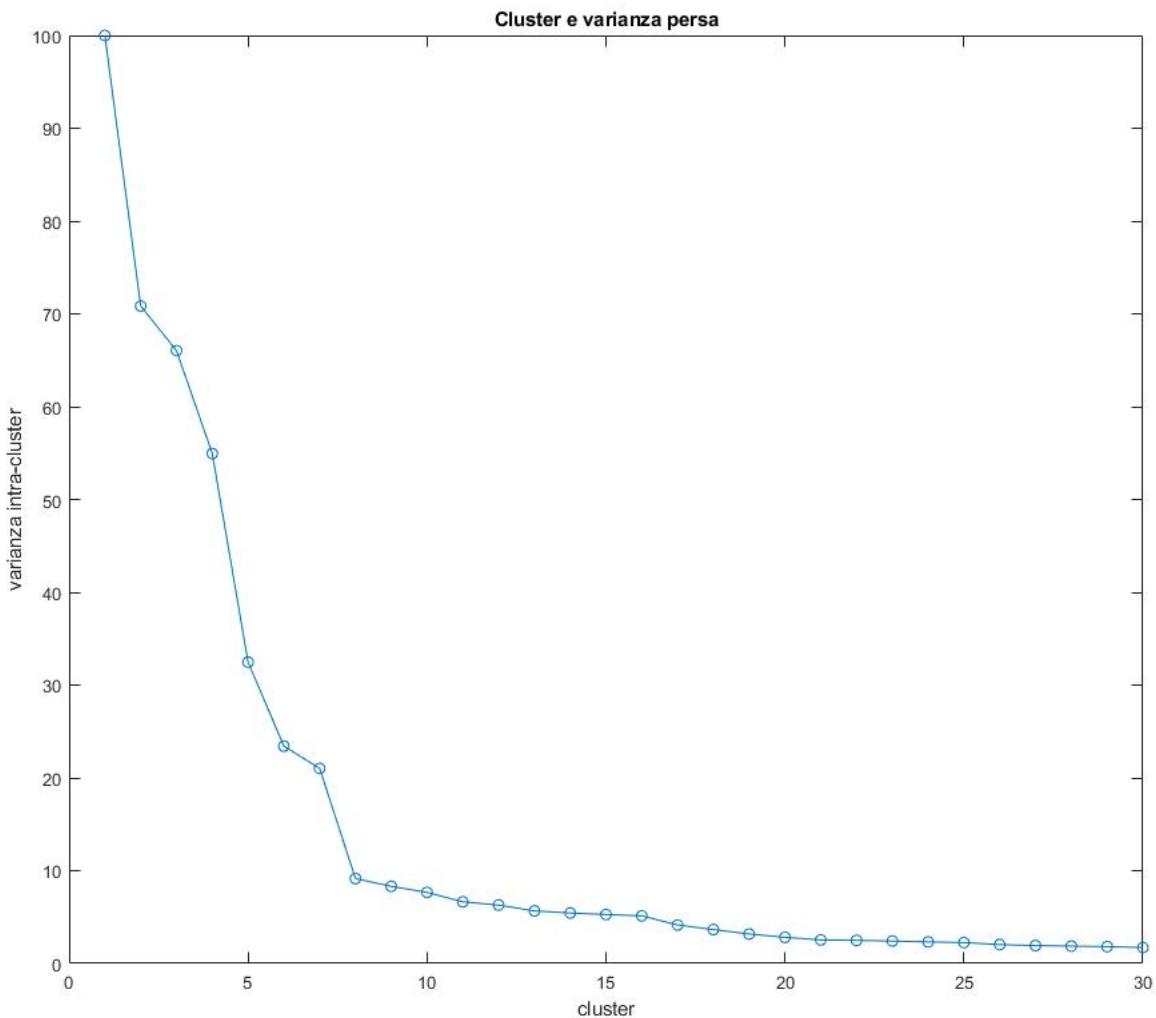


Figura 2.11: Cluster e varianza persa 5 componenti

La tabella successiva riporta quelli che sono i dati ottenuti dall'analisi effettuata scegliendo 5 componenti principali.

5 componenti principali					
Devianza	5 cluster	10 cluster	15 cluster	20 cluster	30 Cluster
Post PCA e Clustering	59.2%	81%	83.1%	85.3%	86.2%
Totale persa	40.7%	18.9%	16.8%	14.6%	13.7%
Post PCA	87.7%	87.7%	87.7%	87.7%	87.7

Tabella 2.1: Analisi devianza 5 componenti

2.3.4.2 PCA & Clustering 6 componenti

Nella figura successiva possiamo notare il dendrogramma generato da JMP dopo la scelta di 6 componenti principali, sapendo che dopo l'operazione di PCA la percentuale di varianza conservata è dell'91.4%.

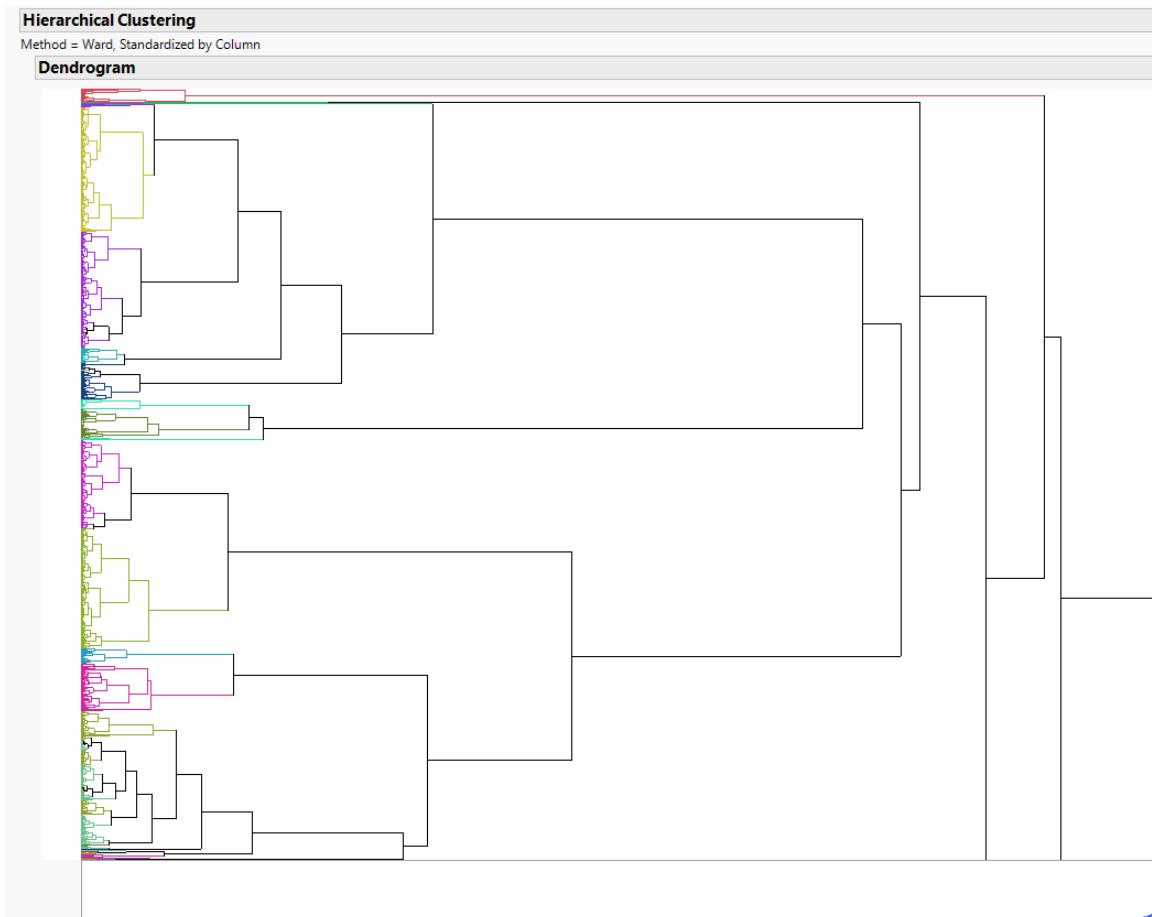


Figura 2.12: Dendrogramma 6 componenti

Per cercare il miglior numero di cluster, si è andati a studiare in maniera approfondita la varianza persa dopo l'operazione di PCA e Clustering attraverso uno script Matlab.

Come possiamo notare nella figura successiva la percentuale di varianza persa si può visualizzare tramite la varianza intra-cluster.

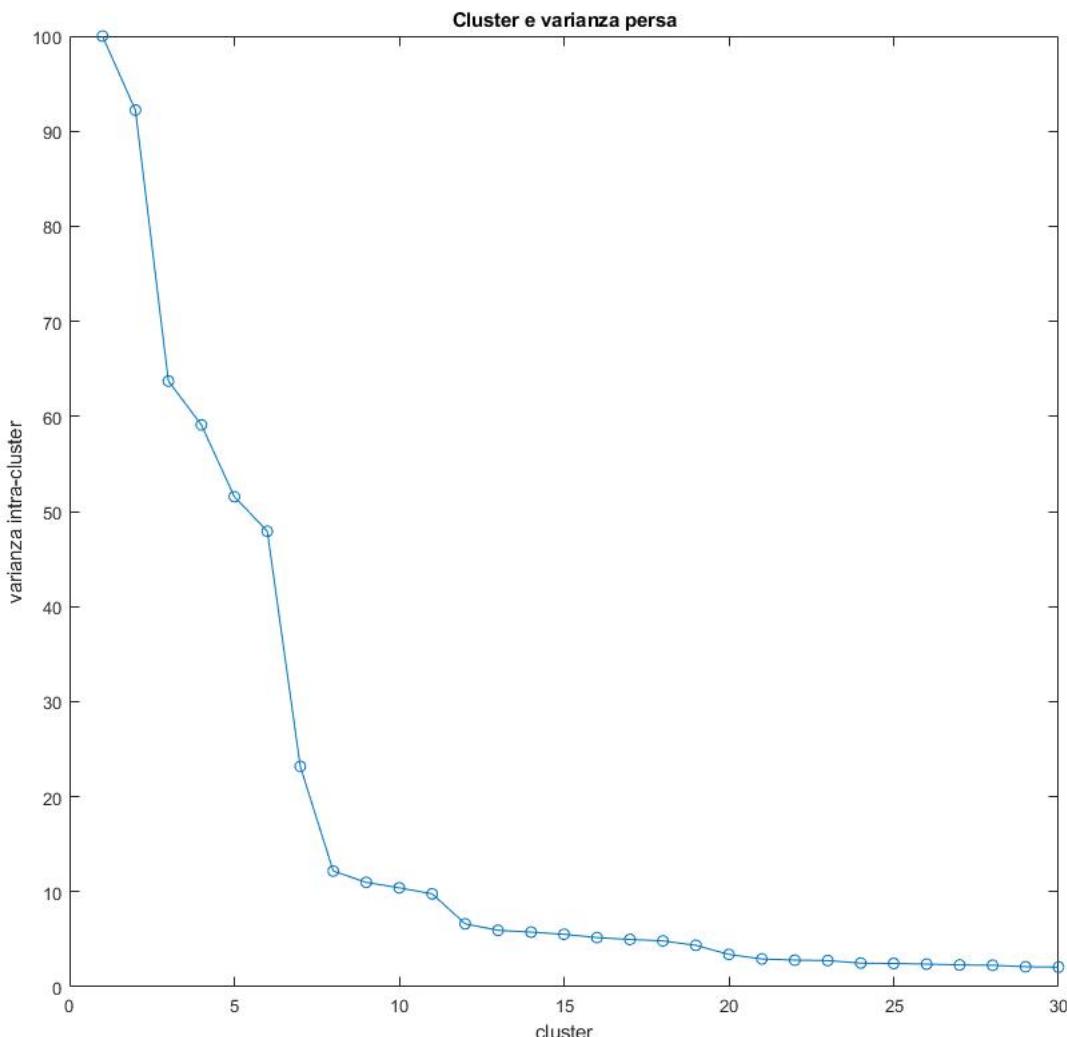


Figura 2.13: Cluster e varianza persa 6 componenti

La tabella successiva riporta quelli che sono i dati ottenuti dall'analisi effettuata scegliendo 6 componenti principali.

6 componenti principali					
Devianza	5 cluster	10 cluster	15 cluster	20 cluster	30 cluster
Post PCA e Clustering	44.2%	81.9%	86.3%	88.3%	89.5%
Totale persa	55.7%	18%	13.6%	11.6%	10.4%
Post PCA	91.4%	91.4%	91.4%	91.4%	91.4%

Tabella 2.2: Analisi devianza 6 componenti

2.3.4.3 PCA & Clustering 7 componenti

Nella figura successiva possiamo notare il dendrogramma generato da JMP dopo la scelta di 7 componenti principali, sapendo che dopo l'operazione di PCA la percentuale di varianza conservata è dell'94.5%.

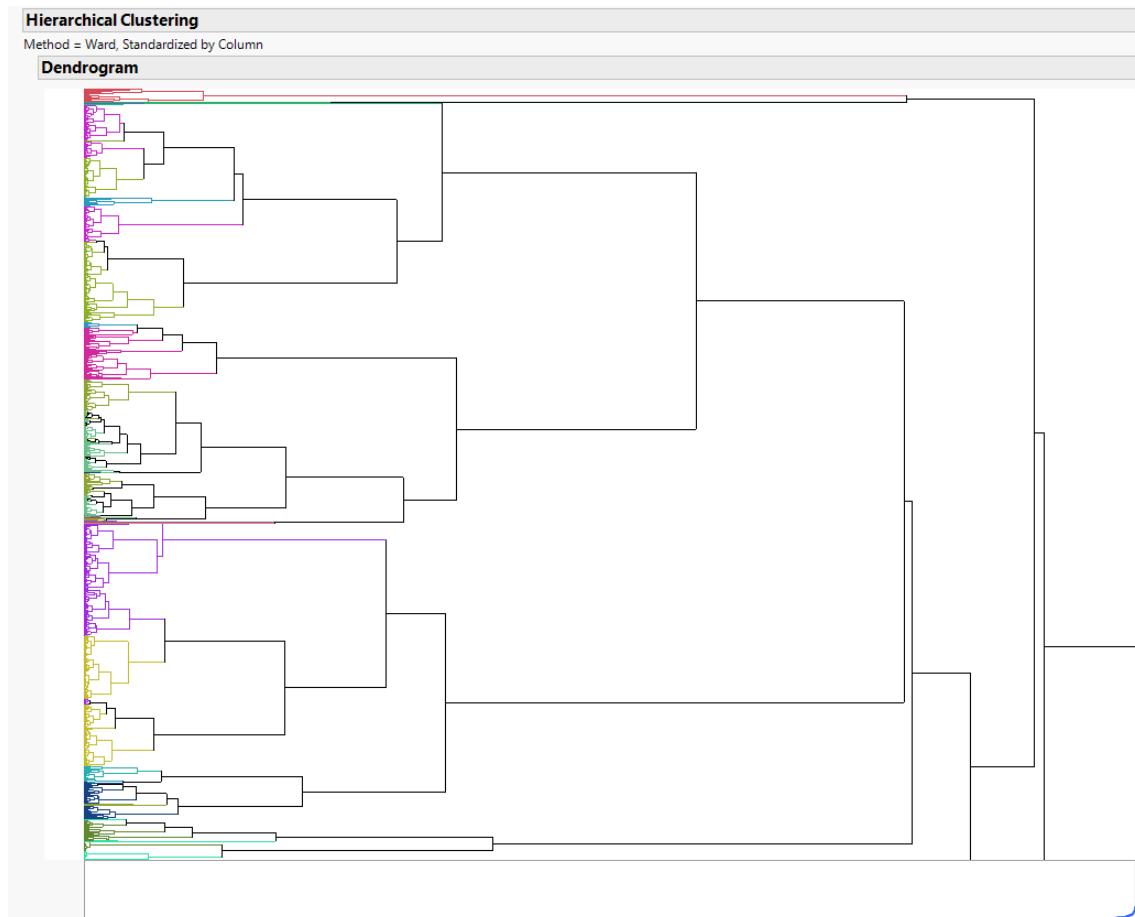


Figura 2.14: Dendrogramma 7 componenti

Per cercare il miglior numero di cluster si è andati a studiare in maniera approfondita la varianza persa dopo l'operazione di PCA e Clustering attraverso uno script Matlab.

Come possiamo notare nella figura successiva la percentuale di varianza persa si può visualizzare tramite la varianza intra-cluster.

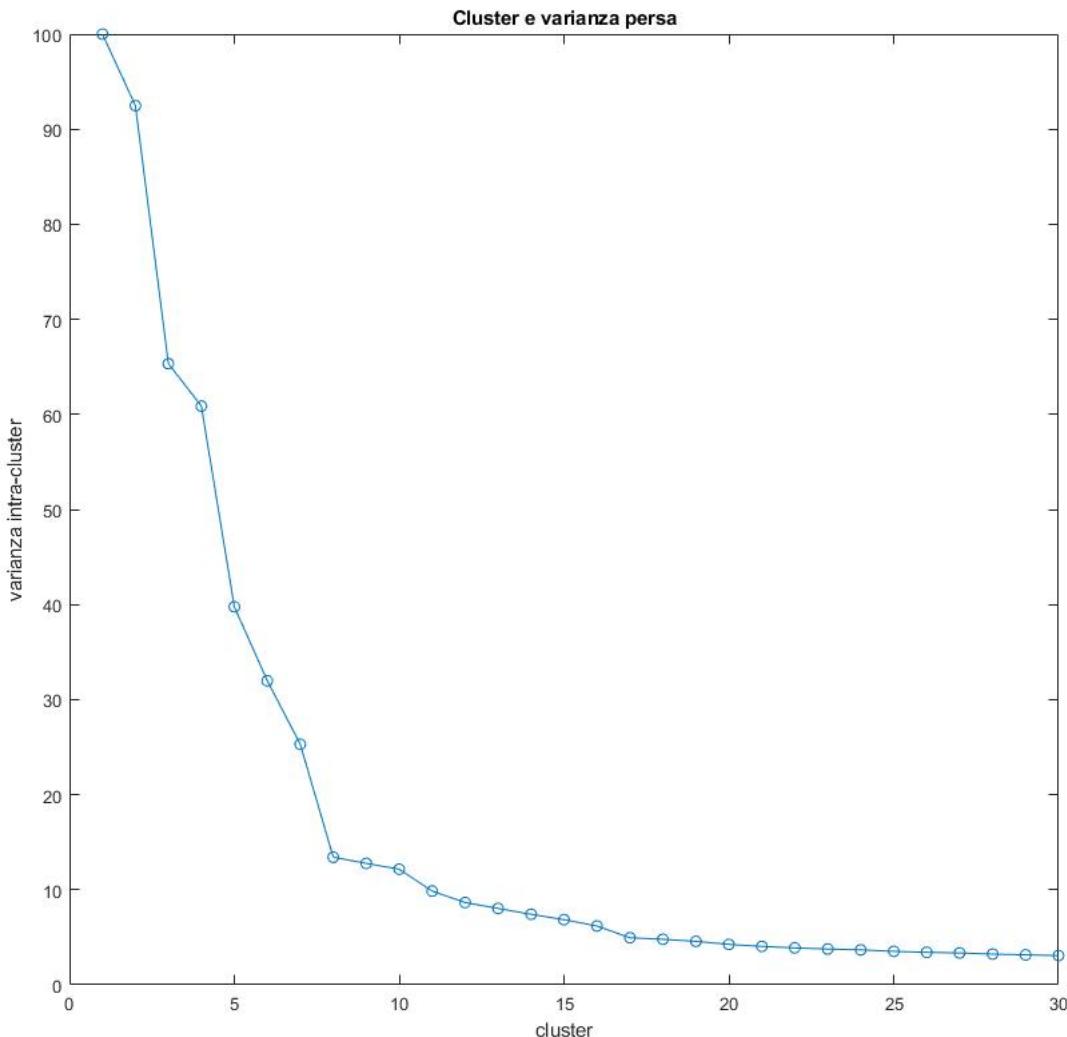


Figura 2.15: Cluster e varianza persa 7 componenti

La tabella successiva riporta quelli che sono i dati ottenuti dall'analisi effettuata scegliendo 7 componenti principali.

7 componenti principali					
Devianza	5 cluster	10 cluster	15 cluster	20 cluster	30 cluster
Post PCA e Clustering	56.9%	83%	88%	90.4%	91.6%
Totale persa	43%	16.9%	11.9%	9.5%	8.3%
Post PCA	94.5%	94.5%	94.5%	94.5%	94.5%

Tabella 2.3: Analisi devianza 7 componenti

2.3.5 Conclusioni

Si va a visualizzare nella figura successiva il grafico risultante delle varie varianze spiegate per ogni PC considerata in precedenza e per differenti cluster.

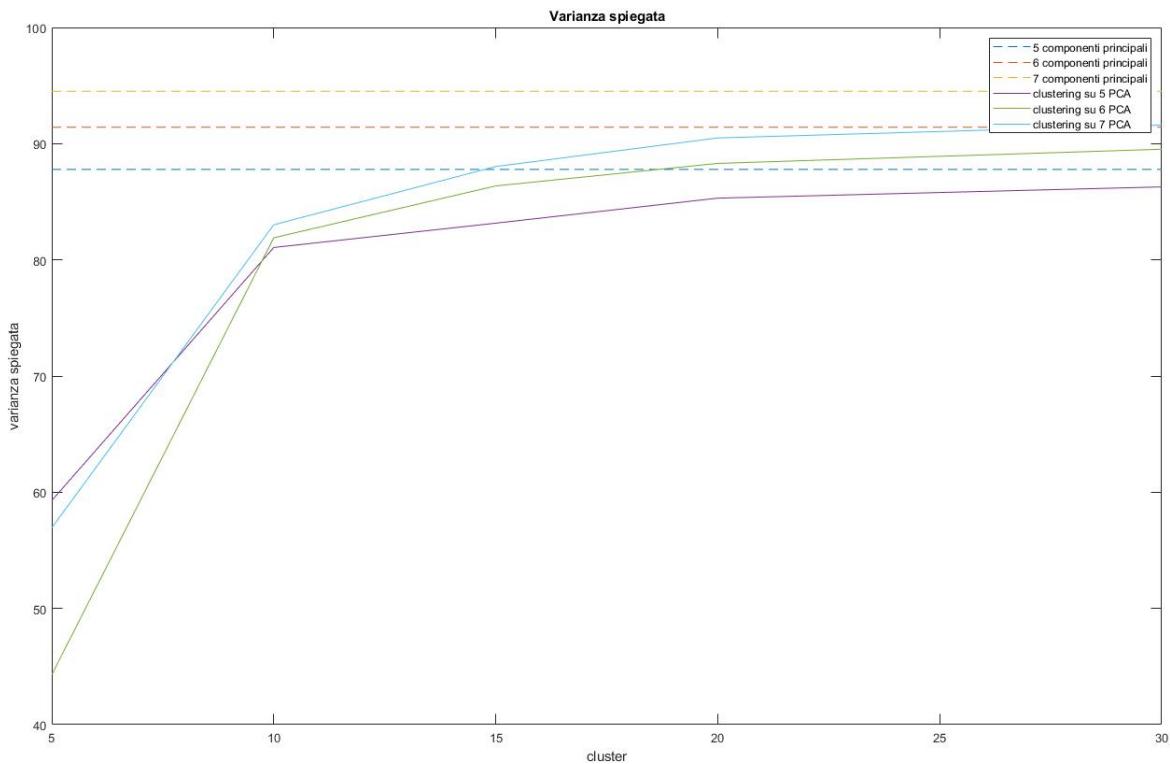


Figura 2.16: Varianza spiegata PCA & Clustering

Percentuale di devianza					
PC number	5 cluster	10 cluster	15 cluster	20 cluster	30 cluster
5	59.2%	81%	83.1%	85.3%	86.2%
6	44.2%	81.9%	86.9%	88.3%	89.5%
7	56.9%	83%	88%	90.4%	91.6%

Tabella 2.4: Analisi devianza PCA & Clustering

A valle dell'analisi condotta, nell'andare a scegliere quello che è il **trade-off**, si è considerata nella scelta l'opzione in grado di ridurre significativamente il numero di osservazioni e di variabili in modo da avere un workload sintetico quanto più simile possibile al workload reale, ossia un workload sintetico avente un'ottima percentuale di varianza spiegata.

Si è scelto quindi di considerare **5 componenti principali** e **15 cluster**, in modo da avere una devianza pari a **83.1%**. Tale scelta viene considerata ottimale in quanto, non essendo a conoscenza del contesto funzionale dell'analisi, il valore di devianza ottenuto viene considerato soddisfacente.

A tal proposito si sottolinea come un aumento in termini di componenti principali o di cluster non comporta un conseguente aumento di devianza spiegata tale da giustificare gli incrementi.

2.3.5.1 Workload sintetico

A valle delle operazioni fatte in precedenza, si sono scelti dei valori randomici da ogni cluster ottenendo il workload sintetico rappresentato in figura.

'VmPmC'	'VmSize'	'VmHVM'	'VmRSS'	'VmPTE'	'Threads'	'MemFree'	'Buffers'	'Cached'	'Active'	'Inactive'	'Dirty'	'Writeback'	'anonPage'	'Mapped'	'Slab'	'PageTables'	'Commiter'	'NumAllocs'	'proc'	'avgThrout'	'avgElapse'	'avgLatency'	'Errors'	Cluster
144012	126424	14224	14224	10	9	5621452	29064	334600	0	131152	300072	5621452	185924	1156	0	67408	27482	26160	6968	167820	510	0	2	1
135844	151436	26612	26648	160	20	5044360	83684	73268	0	365572	578944	5044360	185924	115360	0	126116	28324	9171	7984	347336	2040	0	2	2
170344	170340	31144	31140	200	43	447158	94884	1259568	0	597900	911332	4471580	185924	459632	0	134504	29956	116216	8024	365692	1530	0	2	3
170344	170340	31144	31140	200	52	456560	89400	109572	0	54120	773332	4565604	185924	324400	1760	134500	106788	106788	8024	365264	2040	0	2	4
172388	170340	31948	31948	200	9	494848	89400	109572	0	38800	827200	4948480	185924	107284	0	104152	29956	109200	7608	31956	1000	0	2	5
172388	170340	31948	31948	200	53	494848	89400	1117028	0	47616	8466536	4948480	185924	10352	0	84764	23740	109172	7208	31956	1000	0	2	6
172388	170340	32696	31440	200	9	466892	107840	1117076	0	409280	899404	4668924	185924	152	0	83864	23740	108856	7208	319126	2040	0	2	7
172388	170340	32696	32408	200	9	465516	108308	1117076	0	410360	899404	465516	185924	20	0	84736	23740	108864	7208	314602	1020	0	2	8
192256	185262	43804	37004	252	74	455846	198528	1118124	0	487524	181460	4558460	185924	24	0	89352	23848	113996	7260	331216	510	0	2	9
192256	184808	43804	37864	252	65	455704	198740	1118136	0	488988	918092	4557044	185924	36	0	90264	23848	113532	7480	329112	1020	0	2	10
192256	185262	43804	38024	252	97	457040	198740	1131136	0	489104	181532	4570400	185924	48	0	90468	23848	114116	7260	331596	1530	0	2	11
192256	185262	43804	38024	252	121	455846	198740	1131160	0	489104	181532	4558460	185924	52	0	90468	23848	113596	7260	33096	1020	0	2	12
192256	185264	43804	38820	256	119	455492	197723	1118160	34	491856	16732	4554924	185924	56	0	91304	23848	113596	7252	322276	1530	0	2	13
208184	201016	55500	49548	296	161	454028	203800	1118364	0	514728	905324	4540284	185924	8	0	101888	23848	112676	7304	341520	1020	0	2	14
172388	170340	32696	31460	200	33	465352	132072	1117192	111	412036	921024	465352	185924	164	0	83800	23740	109580	7208	316560	1530	0	2	15

Figura 2.17: Workload sintetico

Capitolo 3

Web Server

3.1 Introduzione

Si vuole effettuare una **performance analysis** di un Web Server attraverso tre fasi:

1. **Capacity Test**: analizzare le condizioni di funzionamento ottimali del sistema andando a ricercare i punti di **knee capacity** e **usable capacity**;
2. **Workload Characterization**: costruire e validare un workload sintetico, ottenuto tramite tecniche di **PCA** e **Clustering** applicate su un workload reale;
3. **Design of Experiment**: studiare l'influenza dei fattori sui tempi di risposta andando a valutare **l'importanza** e la **significatività**.

3.2 Ambiente di lavoro

Il sistema dove il *Web Server Apache 2.4* è stato installato è una macchina virtuale creata con *Oracle VM VirtualBox*, ospitata sulla stessa macchina fisica del client e avente le seguenti caratteristiche:

- Processore: Intel® Core™ i5-5200U CPU @ 2.2GHz
- Numero core: 1
- RAM: 1 GB
- Sistema Operativo: Debian 11 64-bit
- Memoria di massa: 25 GB

La creazione del workload è resa possibile utilizzando il tool *JMeter* nella sua versione 5.5.

3.3 Capacity Test

Il capacity test è una forma di test di performance che ha come obiettivo quello di determinare il limite del sistema in condizioni di carico elevato. Le metriche utilizzate per caratterizzare le performance sono:

- **Response time:** l'intervallo di tempo tra una richiesta di un utente e la risposta del sistema (nel caso in esame, il completamento della risposta);
- **Throughput:** il numero di richieste che possono essere servite per unità di tempo.

3.3.1 Parametri Test

Le risorse utilizzate per effettuare il Capacity Test sono state scelte seguendo tre categorie:

- SMALL, con 2 pagine da 100 e 300 KB;
- MEDIUM, con 2 pagine da 750 e 1000 KB;
- LARGE, con 2 pagine da 1500 e 2000 KB.

Per quanto riguarda il tool **JMeter**, si è costruito il seguente **Test-Plan**:

- Singolo Thread Group con 30 thread;
- Loop count forever;
- Ramp-up period: 30 secondi;
- Durata: 300 secondi;
- Random Controller:
 - HTTP Request Defaults: richieste tutte indirizzate a localhost:8000;
 - Constant Throughput Timer (req/min): 500, 800, 1000, 1300, 1500, 2000, 4000, 8000;
 - HTTP Request: GET di pagine web small, medium e large;
 - Listeners: Summary Report.

3.3.2 Risultati

Al termine delle misurazioni, si sceglie un singolo valore del Throughput e Response Time (secondi) per ogni terna. In particolare si scelgono come metriche la media o la mediana in base al valore dell'indice di COV: se è minore di 0.5 si sceglie la media, altrimenti la mediana.

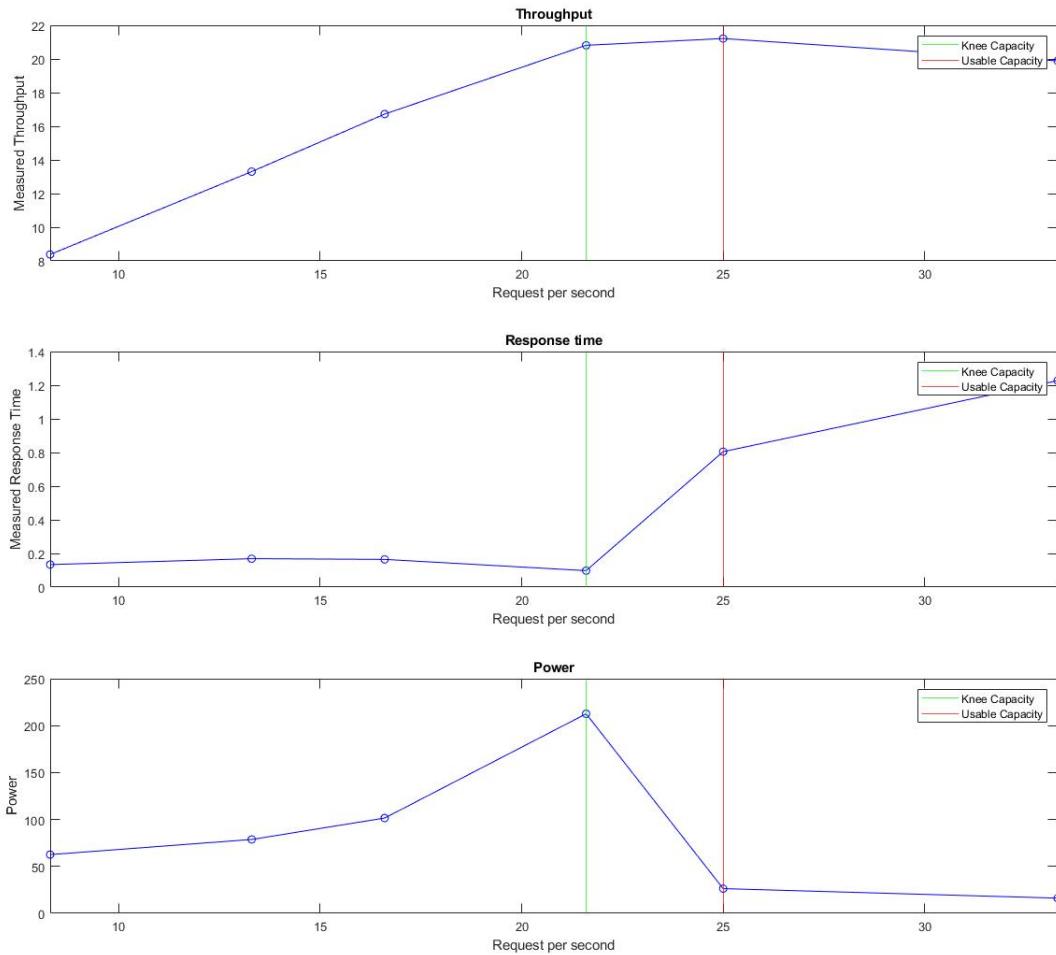


Figura 3.1: Grafici Capacity Test

I valori dopo 33.3 si omettono poiché il Server comincia ad andare a regime per il numero di thread settato, di conseguenza si analizza solo la zona di funzionamento ottimale.

Per ricercare il punto di Knee Capacity si calcola la Potenza che è il rapporto tra il Throughput e il Response Time. Il punto di Usable Capacity si sceglie in base all'andamento del Response Time che inizia ad aumentare rapidamente, come si può osservare anche dall'andamento della Potenza.

Knee Capacity	Usable Capacity
1300 req/min	1500 req/min

Tabella 3.1: Knee Capacity e Usable Capacity

3.4 Workload Characterization

Come già accennato, l'obiettivo della Workload Characterization è costruire e validare un workload sintetico da un workload reale attraverso due tecniche fondamentali: **PCA** e **Clustering**. Le varie

fasi operative sono descritte nella figura successiva.

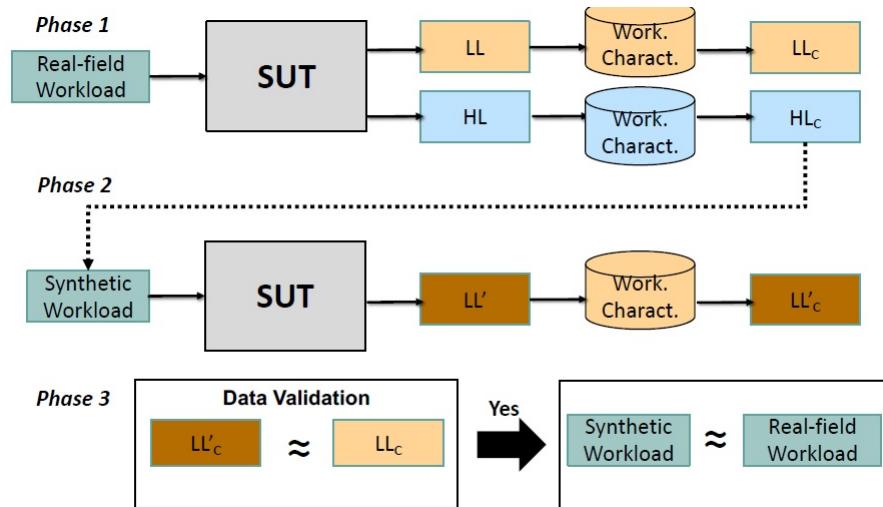


Figura 3.2: Schema Workload Characterization

- **Fase 1:** Inizialmente si sottomette il workload reale al sistema servente, anche in questo caso tramite l'utilizzo di Apache JMeter. Successivamente si collezionano i **parametri di alto livello** e i **parametri di basso livello** in modo da ottenere una caratterizzazione per entrambi i parametri.
- **Fase 2:** Per poter capire se il workload sintetico è rappresentativo di quello reale, si sottomette il workload sintetico di alto livello al sistema servente andando a collezionare e caratterizzare, solo i parametri di basso livello.
- **Fase 3:** Si verifica la bontà della caratterizzazione confrontando i parametri di basso livello del workload reale con quelli del workload sintetico attraverso un test di ipotesi.

Per quanto riguarda i parametri di JMeter, sono stati utilizzati gli stessi del Capacity Test con un Constant Throughput Timer settato a $1000 \frac{\text{req}}{\text{min}}$. Tuttavia, per poter rappresentare un caso realistico, si effettuano richieste verso 40 risorse che vanno da 50KB a 2000KB dividendole sempre nelle categorie SMALL, MEDIUM e LARGE. I parametri di alto livello verranno collezionati mediante un Simple Data Writer mentre quelli di basso livello tramite il comando `vmstat`.

3.4.1 Fase 1

3.4.1.1 Caratterizzazione dei parametri di alto livello

	timeStamp	elapsed	label	responseCode	responseMessage	threadName	dataType	success	bytes	sentBytes	grpThreads	allThreads	URL	Latency	IdleTime	Connect
1	1.680632e+12	300	HTTP Request 9 S...	200	OK	Thread Group 1-1	text	true	461114	126	1	1	http://...	266	0	27
2	1.680632e+12	228	HTTP Request 13...	200	OK	Thread Group 1-1	text	true	1997115	127	1	1	http://...	10	0	0
3	1.680632e+12	11	HTTP Request 1 S...	200	OK	Thread Group 1-1	text	true	51511	125	1	1	http://...	7	0	0
4	1.680632e+12	228	HTTP Request 11...	200	OK	Thread Group 1-1	text	true	1229115	127	1	1	http://...	17	0	0
5	1.680632e+12	259	HTTP Request 8 L...	200	OK	Thread Group 1-1	text	true	1741115	127	2	2	http://...	8	0	0
6	1.680632e+12	211	HTTP Request 6 ...	200	OK	Thread Group 1-2	text	true	973114	126	2	2	http://...	59	0	2
7	1.680632e+12	409	HTTP Request 12...	200	OK	Thread Group 1-1	text	true	1280315	127	2	2	http://...	33	0	0
8	1.680632e+12	759	HTTP Request 14...	200	OK	Thread Group 1-2	text	true	2048315	127	2	2	http://...	32	0	0
9	1.680632e+12	403	HTTP Request 6 L...	200	OK	Thread Group 1-1	text	true	1638715	127	2	2	http://...	25	0	0
10	1.680632e+12	10	HTTP Request 1 S...	200	OK	Thread Group 1-3	text	true	51512	125	3	3	http://...	6	0	2
11	1.680632e+12	200	HTTP Request 13...	200	OK	Thread Group 1-2	text	true	1331515	127	3	3	http://...	13	0	0
12	1.680632e+12	232	HTTP Request 10...	200	OK	Thread Group 1-1	text	true	1843515	127	3	3	http://...	5	0	0
13	1.680632e+12	143	HTTP Request 11...	200	OK	Thread Group 1-2	text	true	563513	126	3	3	http://...	8	0	0
14	1.680632e+12	219	HTTP Request 8 S...	200	OK	Thread Group 1-3	text	true	409913	126	3	3	http://...	14	0	0
15	1.680632e+12	30	HTTP Request 2 S...	200	OK	Thread Group 1-2	text	true	102713	126	3	3	http://...	6	0	0
16	1.680632e+12	54	HTTP Request 2 S...	200	OK	Thread Group 1-3	text	true	102713	126	3	3	http://...	4	0	0
17	1.680632e+12	304	HTTP Request 8 L...	200	OK	Thread Group 1-1	text	true	1741115	127	3	3	http://...	8	0	0
18	1.680632e+12	85	HTTP Request 11...	200	OK	Thread Group 1-1	text	true	1229115	127	3	3	http://...	4	0	0
19	1.680632e+12	28	HTTP Request 9 S...	200	OK	Thread Group 1-1	text	true	461113	126	3	3	http://...	6	0	0
20	1.680632e+12	156	HTTP Request 3 S...	200	OK	Thread Group 1-3	text	true	153913	126	3	3	http://...	18	0	0

Figura 3.3: Parametri alto livello

Il primo passo per la caratterizzazione dei parametri di alto livello è l'**Analisi delle Componenti Principali (PCA)**. Prima di andare a scegliere il numero di componenti principali da utilizzare per la caratterizzazione, si vanno a filtrare gli attributi categorici e gli attributi costanti *ResponseCode* e *idleTime*. Inoltre si nota l'egualanza delle distribuzioni di *allThread* e *grpThread*, di conseguenza una delle due può essere cancellata.

Dunque si può applicare la PCA, ottenendo i seguenti risultati:

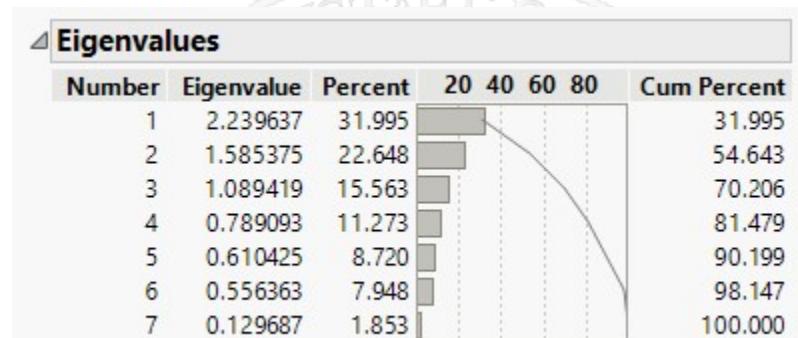


Figura 3.4: Autovalori PCA alto livello

Si è scelto un numero di componenti principali uguale a 6, preservando il 98% di varianza, in modo da non perderne troppa dopo la fase di Clustering.

Il secondo passo per la caratterizzazione è l'operazione di **Clustering**. Si può vedere il dendrogramma generato nella figura successiva:

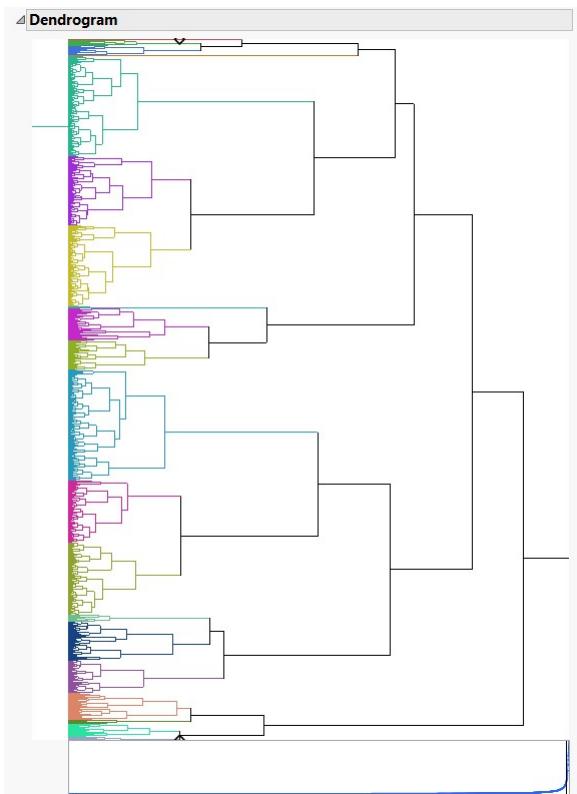


Figura 3.5: Dendrogramma alto livello

Per poter scegliere un valore ottimale, è stato generato un grafico dell'andamento della varianza intra-cluster rispetto al numero di cluster. Dunque si sceglie un valore di 20 cluster, in modo da preservare circa il 78% della varianza spiegata.



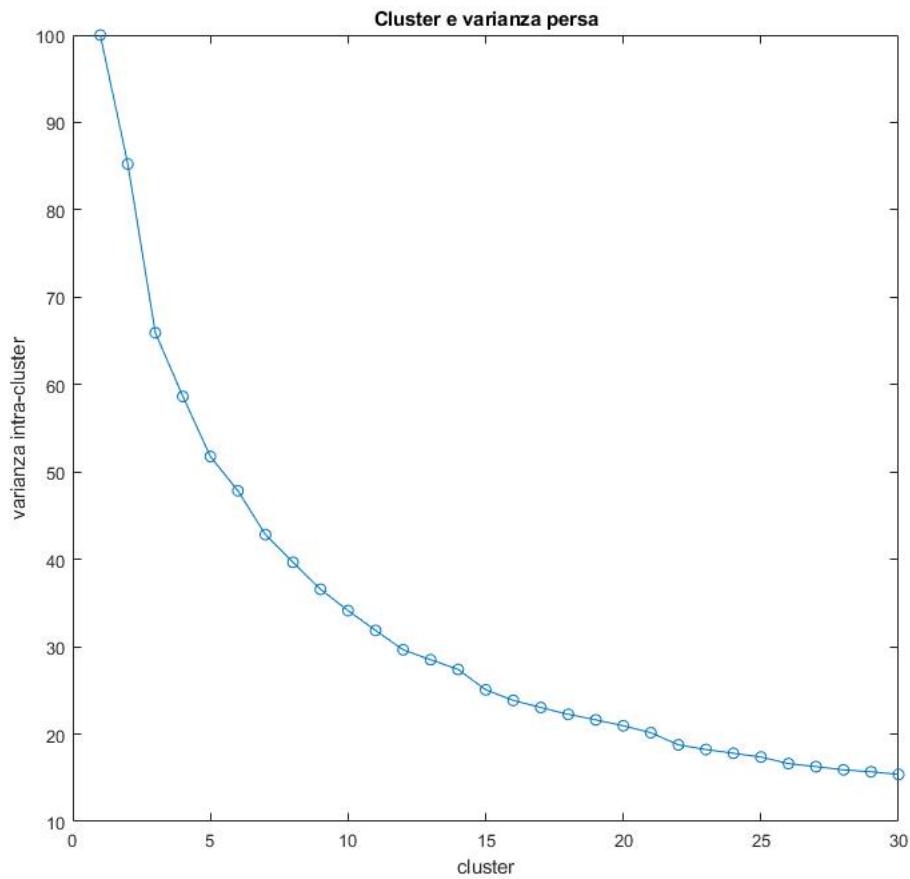


Figura 3.6: Varianza intra-cluster alto livello

Al termine della fase di clustering, si può procedere alla generazione del workload sintetico di alto livello. Per fare ciò, si calcolano per ogni cluster le occorrenze del tipo di richiesta effettuata e si sceglie la più rappresentativa.

Cluster	Request
1	9_S
2	4_S
3	1_L
4	11_S
5	5_S
6	1_S
7	4_M
8	8_S
9	13_L
10	3_M
11	8_M
12	5_L
13	12_M
14	12_L
15	14_L
16	8_L
17	7_S
18	2_M
19	11_L
20	10_M

Tabella 3.2: Occorrenze richieste alto livello

3.4.1.2 Caratterizzazione dei parametri di basso livello

	r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st
1	1	1	97624	244172	12468	243972	63	935	4632	1316	590	1517	9	41	49	2	0
2	4	0	97624	233024	12468	252884	204	0	9100	0	1184	439	0	88	8	4	0
3	4	0	97624	226540	12476	259396	116	0	6616	44	1483	347	0	98	2	0	0
4	4	0	97624	224148	12476	261548	212	0	2356	0	1067	380	2	85	11	2	0
5	3	0	97624	218692	12476	266148	100	0	4716	0	1415	339	0	100	0	0	0
6	1	0	97624	213052	12476	271608	32	0	5488	68	796	342	0	77	21	2	0
7	1	0	97624	205920	12484	277732	52	0	6184	0	660	422	1	97	2	0	0
8	1	0	97624	205104	12492	278640	24	0	924	16	590	304	0	97	3	0	0
9	0	0	97368	203092	12492	280828	40	0	2244	0	438	252	0	91	9	0	0
10	2	0	97368	201588	12492	281756	8	0	908	0	481	251	0	93	7	0	0
11	4	0	97368	200996	12492	281760	16	0	16	0	469	243	0	82	18	0	0
12	1	0	97368	200952	12492	282808	0	0	1052	0	566	292	0	100	0	0	0
13	1	0	97368	193944	12500	284360	0	0	1552	16	557	217	0	88	12	0	0
14	1	0	97368	195324	12500	285216	92	0	944	0	490	337	0	100	0	0	0
15	3	0	97368	197972	12500	285224	0	0	0	0	506	298	0	100	0	0	0
16	1	0	97368	197204	12500	285228	0	0	0	0	814	240	0	80	20	0	0
17	6	0	97368	195444	12500	285228	0	0	0	0	692	219	0	68	32	0	0
18	0	0	97368	198964	12508	285220	0	0	0	16	333	240	0	90	10	0	0
19	2	0	97368	193412	12508	285232	0	0	0	0	562	278	0	95	5	0	0
20	1	0	97368	195540	12508	285232	0	0	0	0	314	298	0	100	0	0	0

Figura 3.7: Parametri basso livello

Anche in questo caso si va a fare filtering prima di applicare le tecniche di **PCA** e **Clustering**: si va a cancellare solo la colonna *st* in quanto costante. In seguito, si può applicare la PCA,

ottenendo i seguenti risultati:

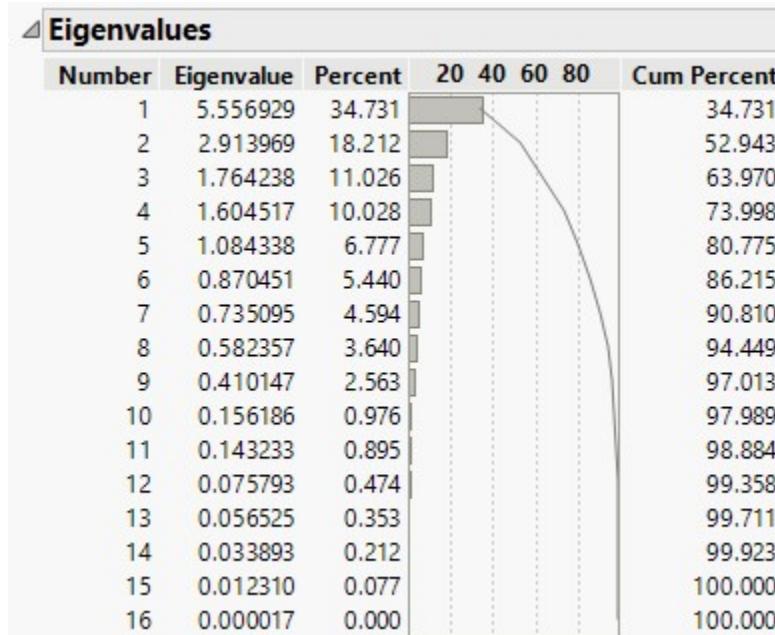
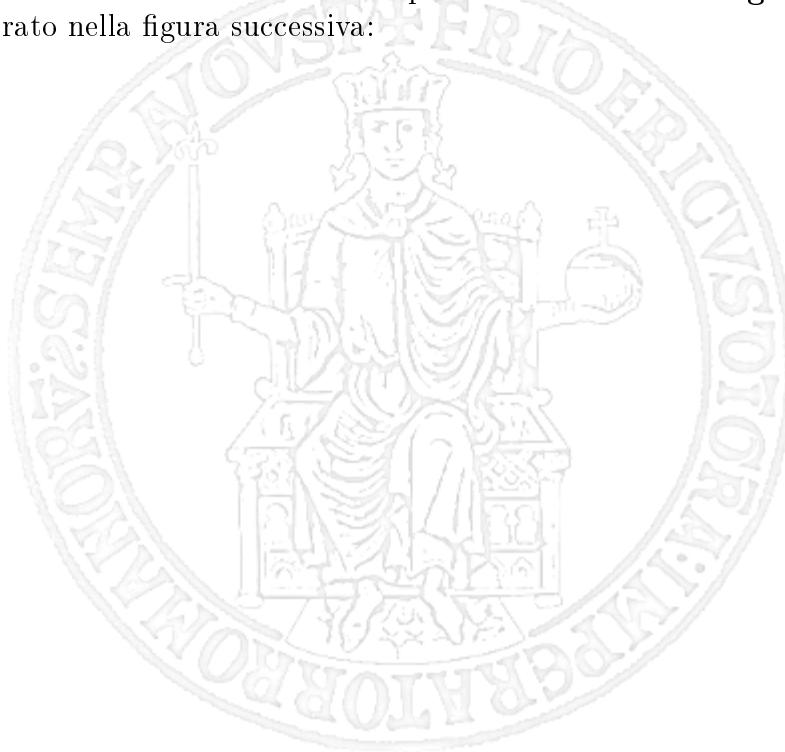


Figura 3.8: Autovalori PCA basso livello

Si è scelto un numero di componenti principali uguale a 9, preservando il 97% di varianza per lo stesso discorso fatto in precedenza.

Il secondo passo per la caratterizzazione è l'operazione di **Clustering**. Si può vedere il dendrogramma generato nella figura successiva:



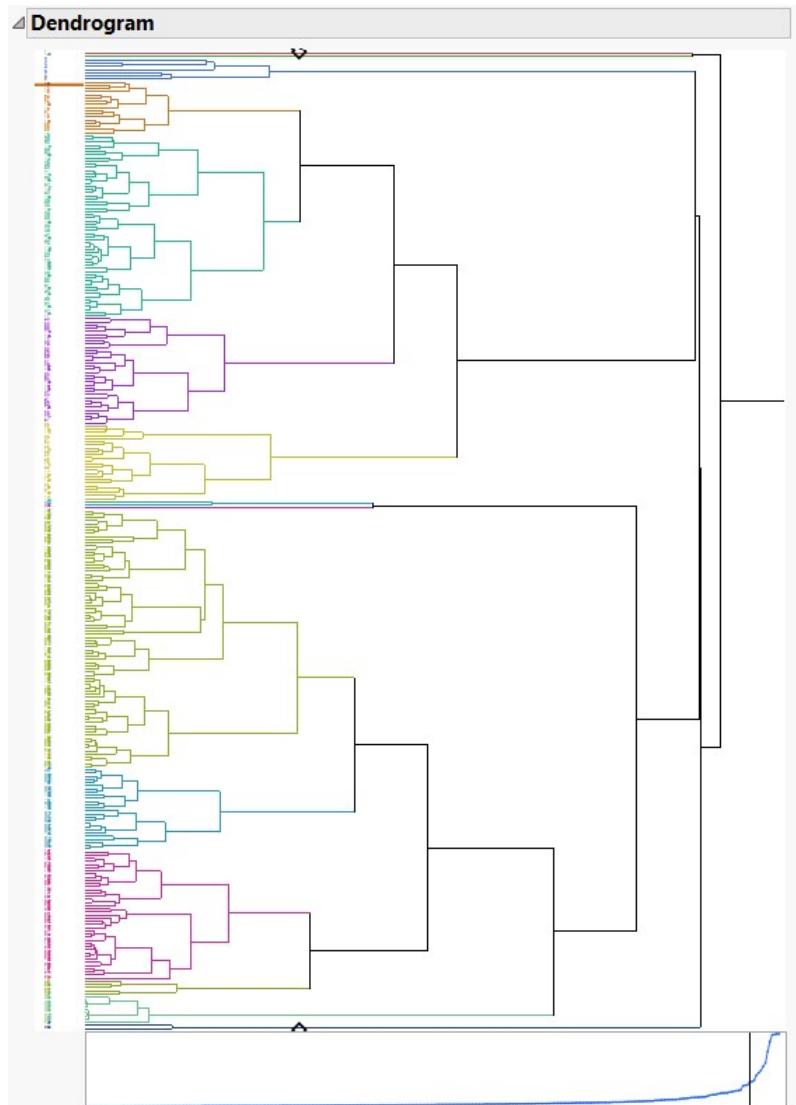


Figura 3.9: Dendrogramma basso livello

Per poter scegliere un valore ottimale, è stato generato, anche in questo caso, un grafico dell'andamento della varianza intra-cluster rispetto al numero di cluster. Dunque si sceglie un valore di 15 cluster, in modo da preservare circa l'84% della varianza spiegata.

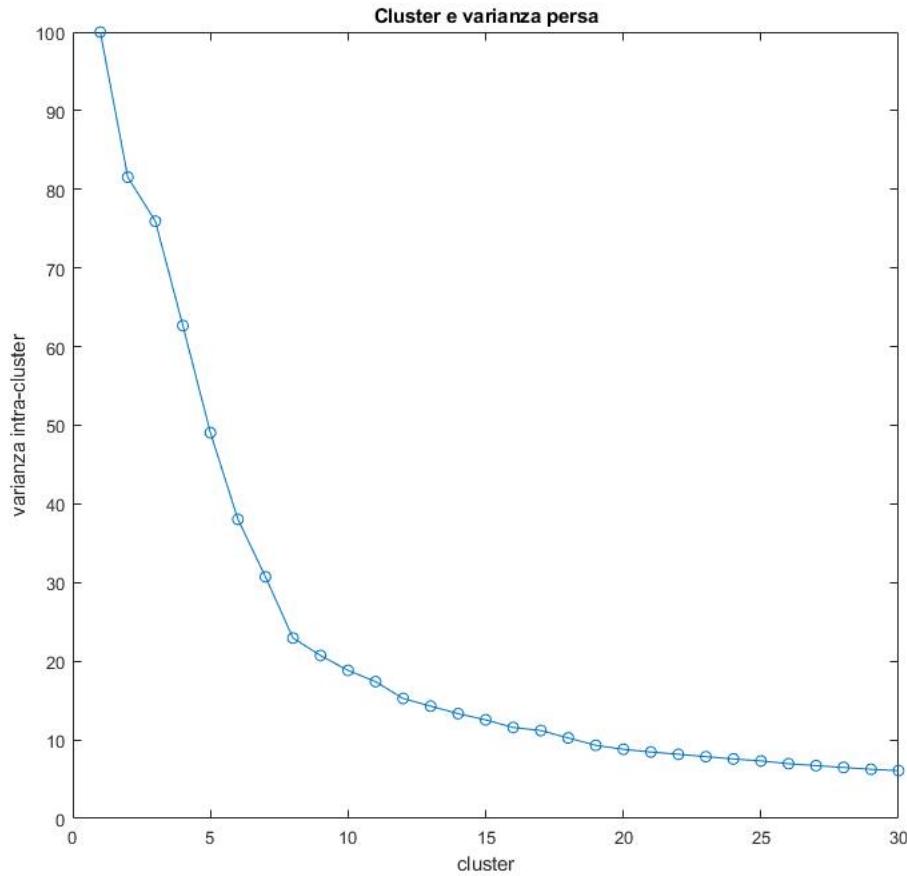


Figura 3.10: Varianza intra-cluster basso livello

Gli elementi per la creazione del workload sintetico in questo caso sono stati scelti casualmente.

3.4.2 Fase 2

3.4.2.1 Caratterizzazione dei parametri di basso livello

In questa fase si sottomette il workload sintetico al sistema servente per poter successivamente collezionare i parametri di basso livello tramite comando *vmstat*. Per rendere possibile il confronto tra i due workload, bisogna prendere lo stesso numero di componenti principali e lo stesso numero di cluster studiati in precedenza. Di conseguenza la scelta è di 9 componenti principali e 15 cluster.

3.4.3 Fase 3

L'ultima fase prevede la validazione tra workload reale e workload sintetico attraverso il confronto statistico tra i parametri di basso livello visti in precedenza. Per poter scegliere il test da effettuare bisogna ragionare inizialmente sulla condizione di normalità delle distribuzioni da analizzare mediante un test visivo. A tale scopo è stato utilizzato uno script MATLAB e grazie alla funzione *qqplot* si può generare, appunto, il grafico Quantile-Quantile:

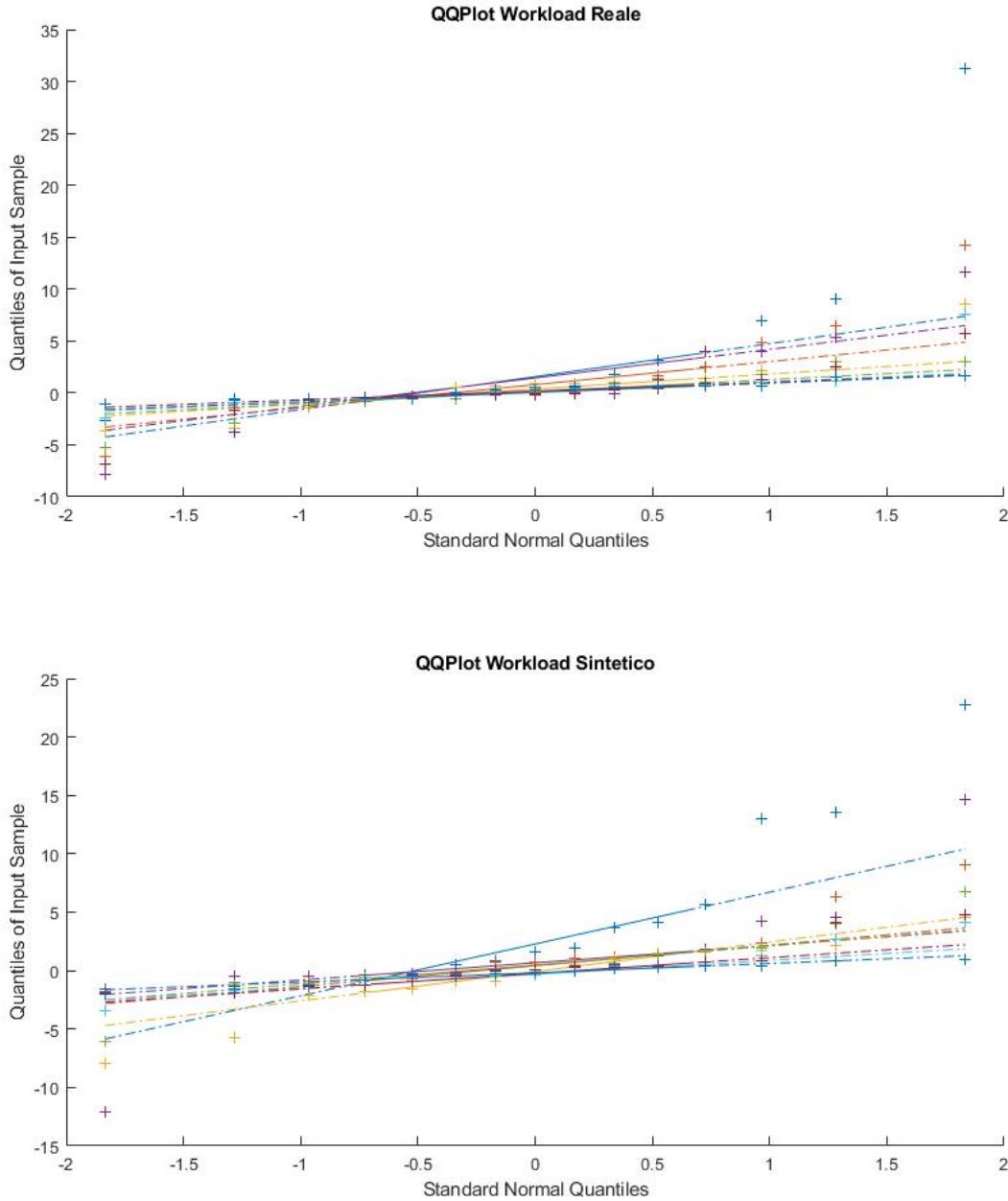


Figura 3.11: QQPlot distribuzioni

Si può notare che non è possibile confermare l'ipotesi di normalità poiché c'è troppa distanza tra i quantili dei campioni e quelli di riferimento della normale.

A valle delle considerazioni fatte, si può applicare un test-non parametrico, in particolare il **test di Wilcoxon** che ha come ipotesi nulla la seguente:

- H_0 : i due campioni indipendenti provengono da distribuzioni di dati con la stessa mediana.

In MATLAB si utilizza la funzione $[p,h] = \text{ranksum}(x,y)$ che genererà un vettore (sia per p che per h) per ogni componente principale.

```
p =
0.7716    0.9339    0.4068    0.5614    0.9669    0.4306    0.8035    0.2134

h =
0    0    0    0    0    0    0    0
```

Figura 3.12: Risultati Test di Wilcoxon

Si può concludere, infine, dicendo che su ogni componente principale **non c'è differenza statistica** e quindi che **il workload sintetico è rappresentativo del workload reale**.

3.5 Design of Experiments

Lo scopo del **Design Of Experiments (DOE)** è quello di mostrare la progettazione di un esperimento caratterizzato da due fattori principali: **Request Rate** o **Intensità**, ossia il numero delle richieste effettuate al minuto e **Page Type** o semplicemente **Type**, ossia la dimensione della pagina richiesta.

Il fattore Request Rate presenta due livelli:

1. 25% della Usable Capacity: $375 \frac{\text{req}}{\text{min}}$;
2. 75% della Usable Capacity: $1125 \frac{\text{req}}{\text{min}}$.

Invece il fattore **Type** presenta quattro livelli:

1. Small: da 50 a 250 KB;
2. Small-Medium: da 500 a 700 KB;
3. Medium-Large: da 1000 a 1200 KB;
4. Large: da 1800 a 2000 KB.

Per ogni combinazione dei fattori, sono state effettuate **cinque ripetizioni da 1 minuto** per un totale di $n = m * k * r = 40$ esperimenti, con m numero di livelli del primo fattore, k numero di livelli del secondo fattore e r numero di ripetizioni. L'esperimento consiste nel misurare il tempo di risposta **Elapsed Time** per tutte le combinazioni dei fattori descritti in precedenza. Dunque, l'obiettivo è quello di verificare l'**importanza** e la **significatività** dei due fattori.

Si possono vedere i risultati degli esperimenti nella figura seguente:

Request rate	Type	Elapsed Time
375	SMALL	25.87281796
375	SMALL	24.07920792
375	SMALL	20.35891089
375	SMALL	20.7970297
375	SMALL	20.72277228
375	SMALL-MEDIUM	78.65432099
375	SMALL-MEDIUM	63.01485149
375	SMALL-MEDIUM	116.8168317
375	SMALL-MEDIUM	55.89851485
375	SMALL-MEDIUM	82.1460396
375	MEDIUM-LARGE	129.8706468
375	MEDIUM-LARGE	130.0297767
375	MEDIUM-LARGE	149.4850746
375	MEDIUM-LARGE	134.1191067
375	MEDIUM-LARGE	131.4950249
375	LARGE	202.9875622
375	LARGE	215.6393035
375	LARGE	235.6144279
375	LARGE	188.7419355
375	LARGE	207.4615385
1125	SMALL	23.61904762
1125	SMALL	23.81038961
1125	SMALL	22.66839827
1125	SMALL	21.06759099
1125	SMALL	20.76450216
1125	SMALL-MEDIUM	85.52211622
1125	SMALL-MEDIUM	88.51302083
1125	SMALL-MEDIUM	86.07899306
1125	SMALL-MEDIUM	94.12413194
1125	SMALL-MEDIUM	83.61265165
1125	MEDIUM-LARGE	196.5474517
1125	MEDIUM-LARGE	184.3484043
1125	MEDIUM-LARGE	177.5994741
1125	MEDIUM-LARGE	159.7585302
1125	MEDIUM-LARGE	160.6179286
1125	LARGE	1611.650649
1125	LARGE	1818.955862
1125	LARGE	1865.256267
1125	LARGE	1715.068306
1125	LARGE	2028.036072

Figura 3.13: Esperimenti DoE

3.5.1 Valutazione del modello

A questo punto si va a valutare il modello regressivo generato tramite il software JMP. Dall'*Analisi della Varianza*, che fornisce le **Sum of Square** del modello e dell'errore, si possono calcolare le percentuali di varianza spiegata.

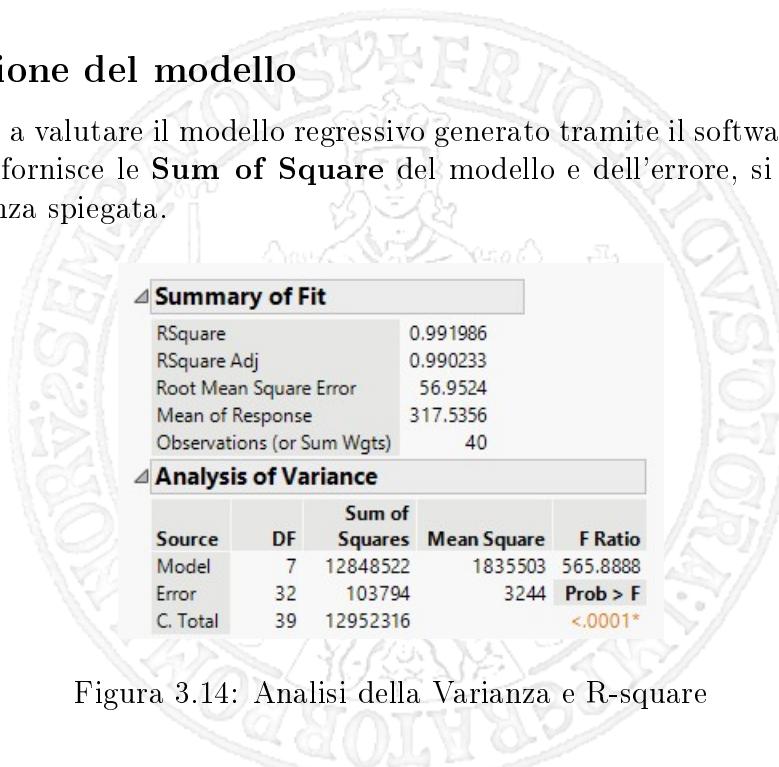


Figura 3.14: Analisi della Varianza e R-square

$$\frac{SS_{modello}}{SS_{totale}} = \frac{12848522}{12952316} = 99.19\%$$

$$\frac{SS_{errore}}{SS_{totale}} = \frac{103794}{12952316} = 0.80\%$$

Si nota che il primo valore coincide con l'R-Square descritto in figura poiché anche quest'ultimo può essere utilizzato per valutare la bontà del modello regressivo, tuttavia sono stati mostrati entrambi per una questione di completezza.

3.5.2 Importanza dei fattori

Considerata la SS del modello, si può studiare l'**importanza** dei fattori e della loro interazione attraverso il rapporto tra la SS dei casi esaminati e la SS totale.

Effect Tests					
Source	Nparm	DF	Sum of Squares	F Ratio	Prob > F
Request rate	1	1	1694892.4	522.5383	<.0001*
Type	3	3	6462545.8	664.1381	<.0001*
Request rate*Type	3	3	4691083.8	482.0898	<.0001*

Figura 3.15: Test degli effetti

$$\frac{SS_{responseRate}}{SS_{totale}} = \frac{1694892}{12952316} = 13.08\%$$

$$\frac{SS_{type}}{SS_{totale}} = \frac{6462545}{12952316} = 49.89\%$$

$$\frac{SS_{responseRate*type}}{SS_{totale}} = \frac{4691083}{12952316} = 36.21\%$$

Come si può vedere il fattore più importante è Type che spiega circa il 50% della varianza mentre Response Rate e il fattore d'interazione spiegano rispettivamente circa il 13% e il 36% della varianza.

3.5.3 Significatività dei fattori

Per studiare la **significatività** dei fattori sarà necessario determinare il tipo di *ANOVA (ANalysis Of VAriance)* da utilizzare. Per fare ciò si andranno a valutare le condizioni di normalità e di omoschedasticità. Grazie sempre all'utilizzo del software JMP, è possibile salvare in colonna i **residui**.

3.5.3.1 Normalità

Bisogna verificare se gli **errori** sono approssimabili a una distribuzione normale. Per fare ciò si effettua un test visivo attraverso il grafico Quantile-Quantile.

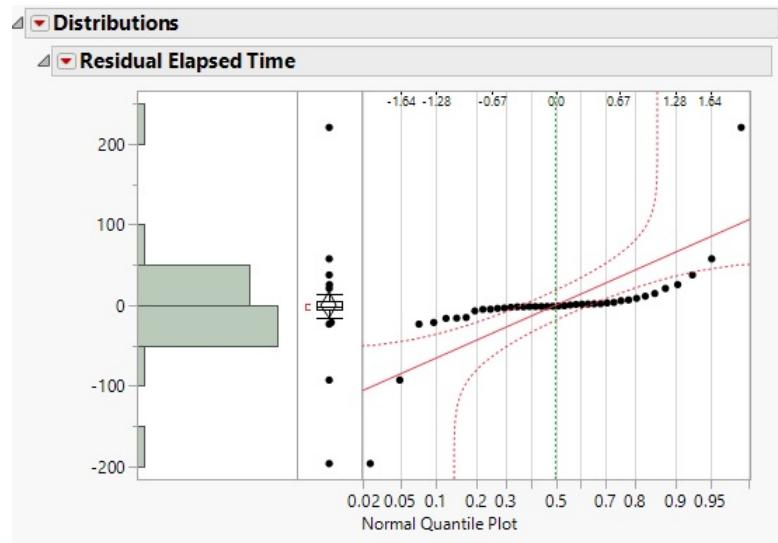


Figura 3.16: QQPlot residui

Senza effettuare ulteriori test, si può subito confermare la **non normalità** dei dati poiché non seguono in maniera evidente la curva di confidenza della normale.

3.5.3.2 Omoschedasticità

Anche in questo caso si effettua un test visivo tramite il diagramma dei residui di Elapsed Time rispetto ai previsti.

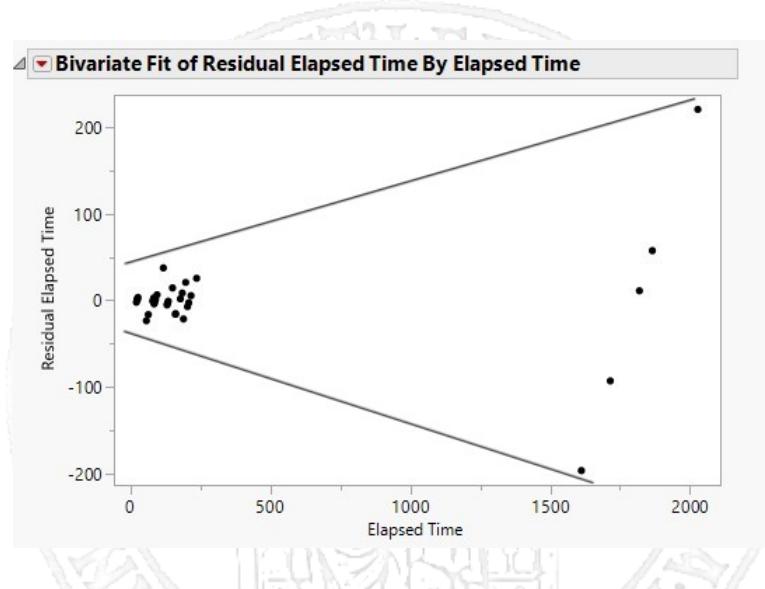


Figura 3.17: Diagramma residui-previsti

Si può notare la presenza di un **trend** e dunque si può già dire che i dati risultano **non omoschedastici**. Tuttavia per avere ulteriore conferma, si effettua il test di O'Brien che ha come ipotesi nulla l'uguaglianza delle varianze.

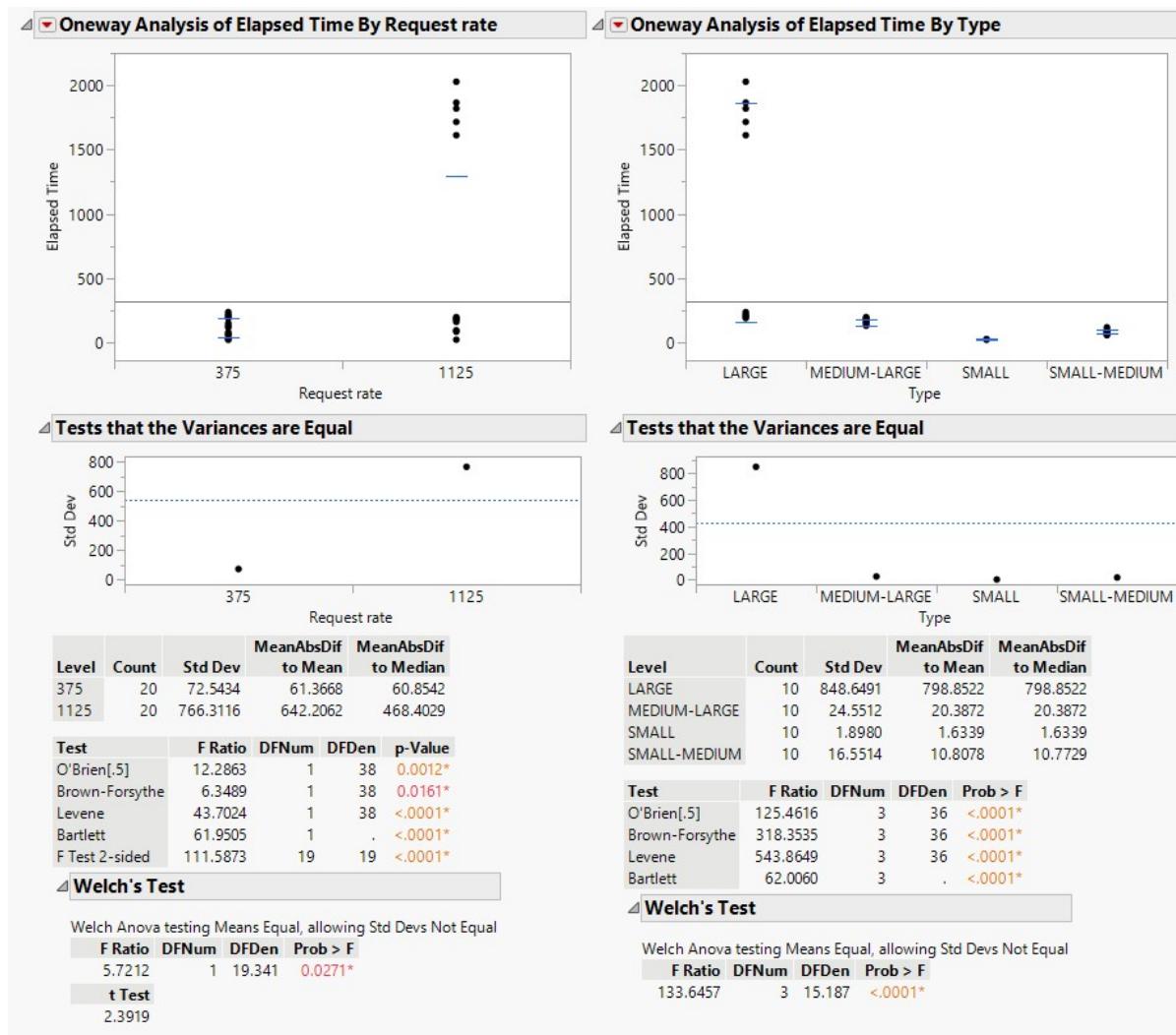


Figura 3.18: Test di O'Brien

Come si può vedere dai valori dei p-value, l'ipotesi nulla viene rigettata quindi non c'è uguaglianza delle varianze e di conseguenza non c'è omoschedasticità.

3.5.3.3 Test ANOVA

Poiché i dati risultano **non normali** e **non omoschedastici**, per valutare la significatività bisogna applicare un test non parametrico, in questo caso il **test di Wilcoxon/Kruskal-Wallis**.

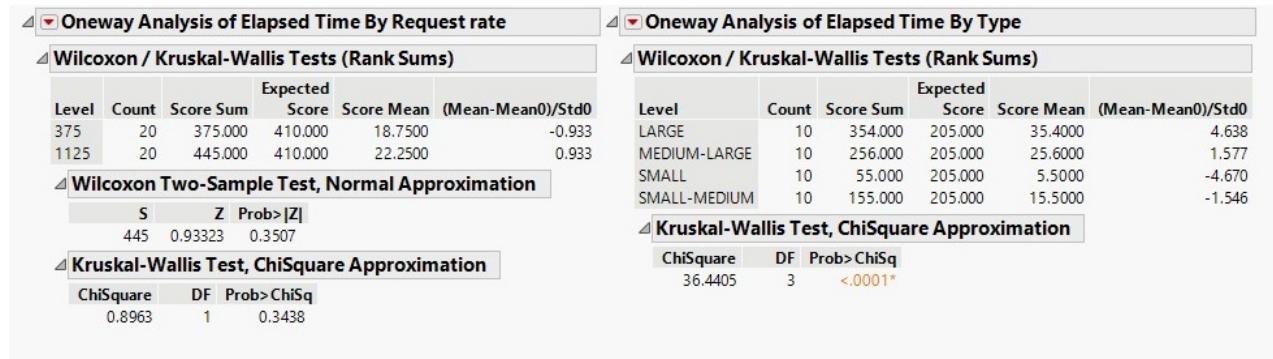


Figura 3.19: Test di Wilcoxon/Kruskal-Wallis

Nel caso di Request Rate, l'ipotesi nulla non viene rigettata quindi il fattore **non può essere considerato statisticamente significativo**. Nel caso di Type avviene l'opposto, ossia il p-value indica che l'ipotesi H_0 è rigettata e di conseguenza il fattore è **statisticamente significativo**.



Capitolo 4

Dependability

4.1 Esercizio 1

4.1.1 Traccia

Trovare la $R(t)$ e l'MTTF per il sistema di cui viene fornito l'RBD. Nel calcolo dell'MTTF, assumere che tutti i componenti siano identici e falliscano randomicamente con failure rate λ .

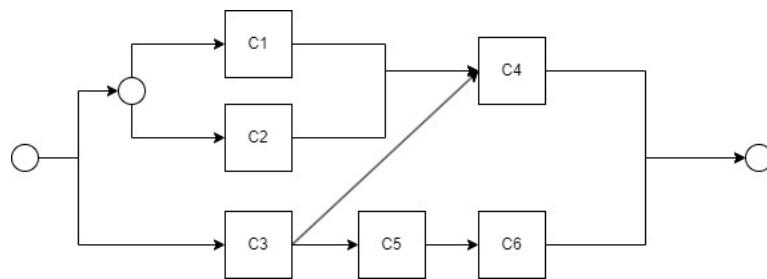


Figura 4.1: RBD

4.1.2 Soluzione

Il sistema è composto da sei componenti con la stessa Reliability R , dunque:

$$R_i(t) = e^{-\lambda t} \text{ con } i = 1..6$$

Dalla figura possiamo notare un parallelo tra C_1 e C_2 e una serie tra C_5 e C_6 quindi lo schema può essere semplificato in questo modo:

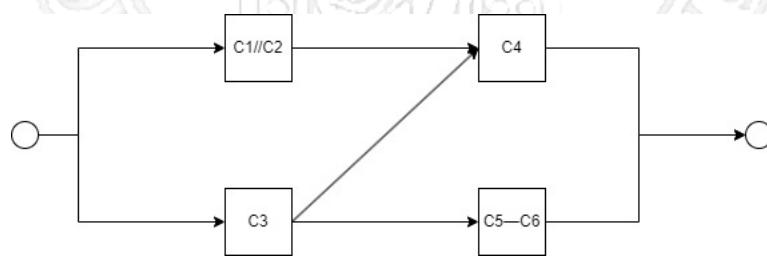


Figura 4.2: RBD semplificato

Dove:

$$R_{1//2} = 1 - (1 - R)^2 \quad R_{5-6} = R * R = R^2$$

4.1.2.1 Teorema dell'Upper Bound

Il teorema dell'upper bound definisce un limite superiore alla Reliability del sistema calcolando il parallelo dei **success path**. Si modifica l'RBD in questo modo:

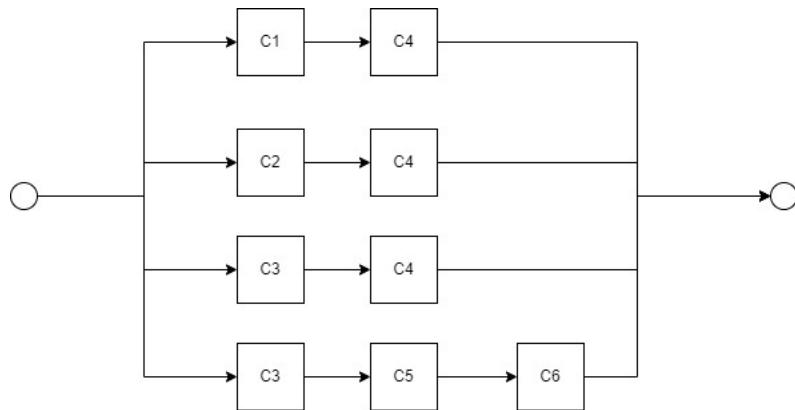


Figura 4.3: RBD con success path

Si calcolano le Reliability delle serie:

$$R_{1-4} = R^2$$

$$R_{2-4} = R^2$$

$$R_{3-4} = R^2$$

$$R_{3-5-6} = R^3$$

La Reliability totale avrà come Upper Bound il valore che si ricava dai paralleli:

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

4.1.2.2 Conditioning

Si può procedere ora con il conditioning, ossia possiamo stabilire che uno dei componenti, tipicamente quello più critico per la nostra analisi, sia fallito. Otteniamo così da un lato un sistema S1 che prevede l'assenza di quel blocco e dall'altro un sistema S2 in cui quel blocco ha sempre successo. La probabilità che il sistema non fallisca, secondo Bayes, è pari alla probabilità che il sistema S1 non fallisca, più la probabilità che il sistema S2 non fallisca. Si procede a condizionare il sistema rispetto a **C4**, calcolando la reliability del sistema e garantendo che sia minore dell'upper bound visto in precedenza.

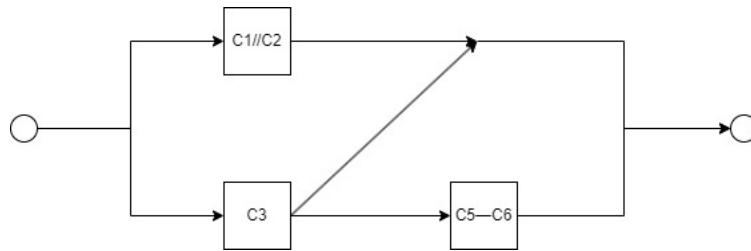


Figura 4.4: Sistema con C4 funzionante

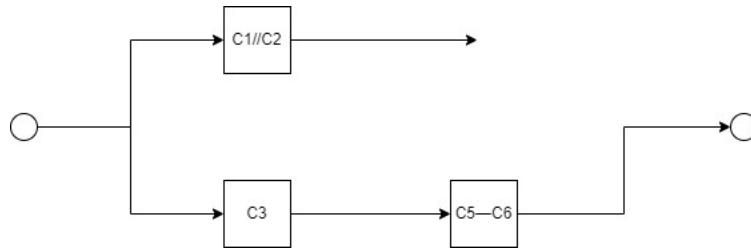


Figura 4.5: Sistema con C4 fallito

Si calcola la Reliability totale con il teorema di Bayes:

$$R_{sys} = R * P(\text{SistemaFunzionante}|C4\text{Funziona}) + (1 - R) * P(\text{SistemaFunzionante}|C4\text{NonFunziona})$$

Quando C4 è funzionante, si ha che il parallelo tra C1 e C2 è in parallelo con C3, mentre quando C4 fallisce, si ha che C3 è in serie con la serie tra C5 e C6.

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

4.1.2.3 MTTF

Avendo calcolato la Reliability dell'intero sistema, si può passare al calcolo del MTTF sia nel caso del teorema dell'Upper Bound sia nel caso del teorema di Bayes:

$$\begin{aligned} MTTF_{UB} &= \int_0^{\infty} (-R^9 + 3R^7 + R^6 - 3R^5 - 3R^4 + R^3 + 3R^2) dt = \\ &= \int_0^{\infty} (-e^{-9\lambda t} + 3e^{-7\lambda t} + e^{-6\lambda t} - 3e^{-5\lambda t} - 3e^{-4\lambda t} + e^{-3\lambda t} + 3e^{-2\lambda t}) dt = \frac{1219}{1260\lambda} \end{aligned}$$

$$MTTF_{bayes} = \int_0^{\infty} (-2R^4 + 4R^3 - 2R^2 + R) dt = \int_0^{\infty} (-2e^{-4\lambda t} + 4e^{-3\lambda t} - 2e^{-2\lambda t} + e^{-\lambda t}) dt = \frac{5}{6\lambda}$$

Confrontando i risultati ottenuti risulta che il risultato calcolato con la regola di Bayes è inferiore al limite superiore.

$$MTTF_{bayes} = \frac{5}{6\lambda} < \frac{1219}{1260\lambda} = MTTF_{UB}$$

4.2 Esercizio 2

4.2.1 Traccia

Si vogliono confrontare due differenti schemi usati per incrementare la reliability di un sistema usando la ridondanza. Supponiamo che il sistema abbia bisogno di componenti identici in serie per funzionare correttamente. Supponiamo, inoltre, che ci siano dati ($m \times s$) componenti. Dato che la reliability di un singolo componente è r , derivare le espressioni per la reliability di due configurazioni. Per $m=5$ e $s=3$, confrontare le due espressioni in funzione di un mission time t . L'MMTF del componente è di 1400 ore. Dei due schemi in figura, quale fornisce la più alta reliability? Modificare lo schema che ha minore reliability per raggiungere la stessa dell'altro dato l'MTTF in sovrapposizione (1440 ore).

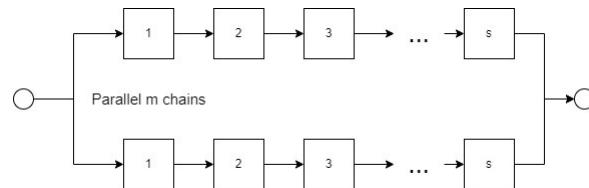


Figura 4.6: Parallelo di serie

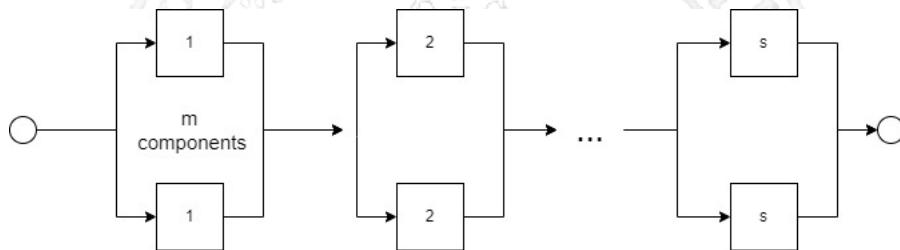


Figura 4.7: Serie di paralleli

4.2.2 Soluzione

Nelle figure sono presenti due RBD, un parallelo di serie e una serie di paralleli. Ciò che si vuole capire è quale dei due garantisce una Reliability più alta. Visivamente, si può dare già dare una risposta considerando i success path. Si noti che i success path del primo RBD sono solamente m mentre quello del secondo sono m^s , pertanto il secondo è più reliable rispetto al primo. Si va a dimostrare quanto detto calcolando la Reliability di entrambi gli RBD:

$$R_{sysA} = 1 - (1 - r^s)^m$$

$$R_{sysB} = (1 - (1 - r)^m)^s$$

Sostituendo nelle espressioni $m = 5$ e $s = 3$ otteniamo:

$$R_{sysA} = 1 - (1 - r^3)^5$$

$$R_{sysB} = (1 - (1 - r)^5)^3$$

Sapendo che l'indice MTTF dei componenti risulta pari a 1400 ore, si assume una reliability per ciascun componente come una distribuzione esponenziale:

$$r = e^{-\frac{t}{1400}}$$

4.2.2.1 Confronto

A questo punto, si possono mettere in relazione i due sistemi utilizzando il software MATLAB in modo da sovrapporre i grafici degli andamenti della Reliability. Nel primo caso abbiamo l'andamento della Reliability del sistema rispetto alla Reliability del singolo componente, nel secondo invece rispetto al tempo (in ore).



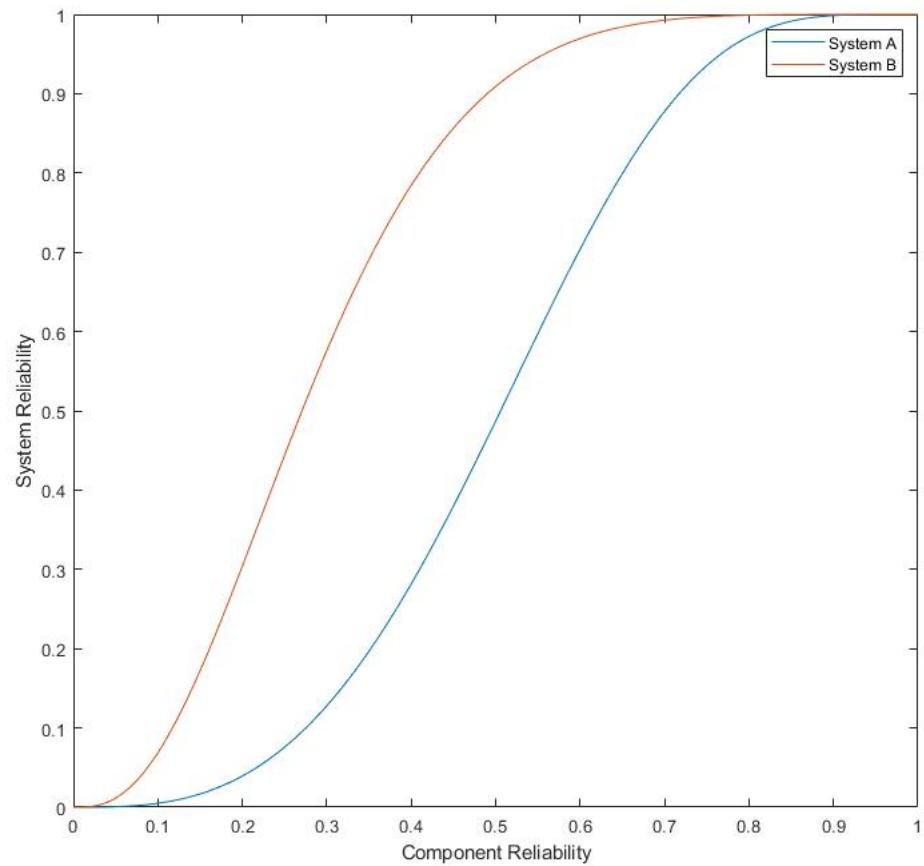
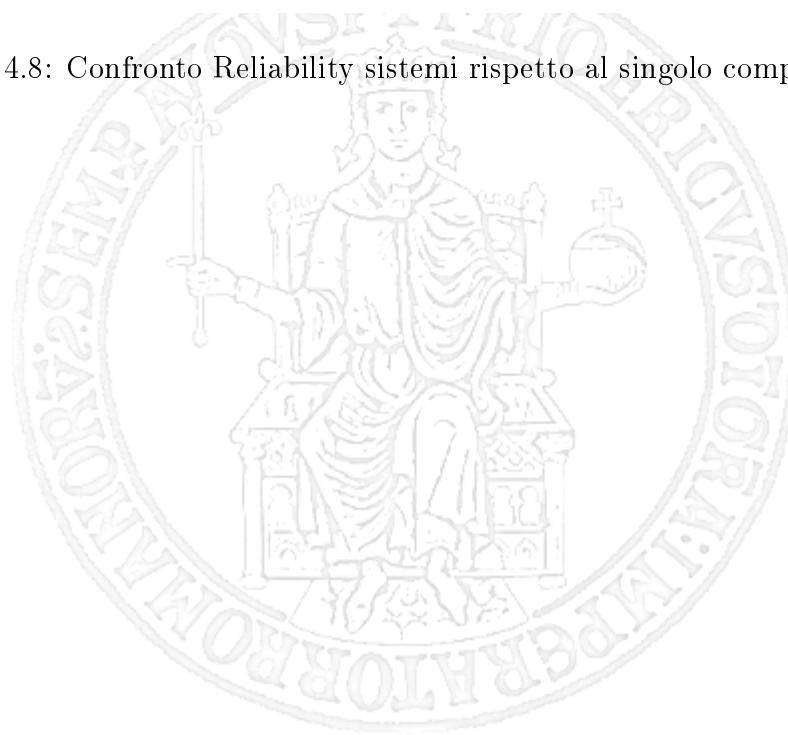


Figura 4.8: Confronto Reliability sistemi rispetto al singolo componente



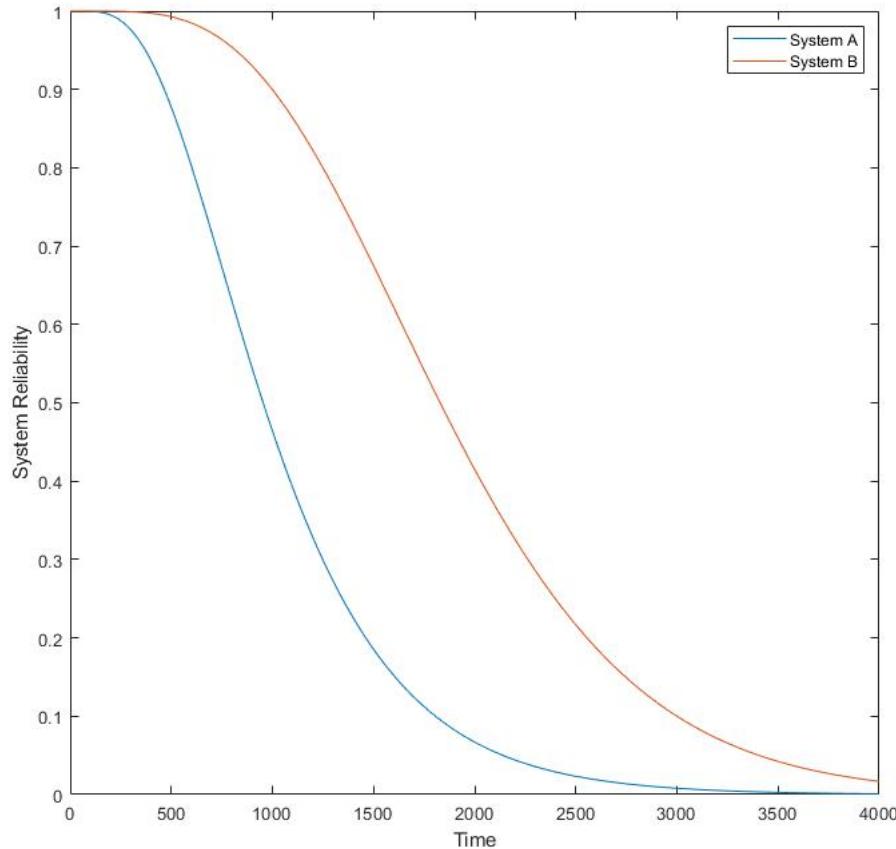


Figura 4.9: Confronto Reliability sistemi rispetto al tempo

Come si evince dal grafico, la Reliability della serie di paralleli è sempre maggiore di quella del parallelo di serie, sia a parità di reliability del singolo componente, sia per qualsiasi valore di mission time.

4.2.2.2 Equilibrio dei due sistemi

Per garantire la stessa reliability per uno specifico mission time, bisogna aumentare la reliability del sistema A uguagliandola con quella del sistema B. Per farlo, **si aumenta il numero m di paralleli**.

Per trovare il valore di m necessario si calcola la Reliability del sistema B imponendo $t = 1400$ h (mission time assegnato):

$$R_{sysB}(1400) = 0.7267$$

Dunque:

$$R_{sysA} = 1 - (1 - r^3)^m = 0.7267$$

$$= (1 - e^{-3}) = 0.2733$$

$$\begin{aligned}
 &= (0.9502)^m = 0.2733 \\
 &= m \log(0.9502) = \log(0.2733)
 \end{aligned}$$

Quindi il valore finale sarà:

$$m = \frac{\log(0.2733)}{\log(0.9502)} \approx 26$$

Si può avere una conferma anche graficamente:

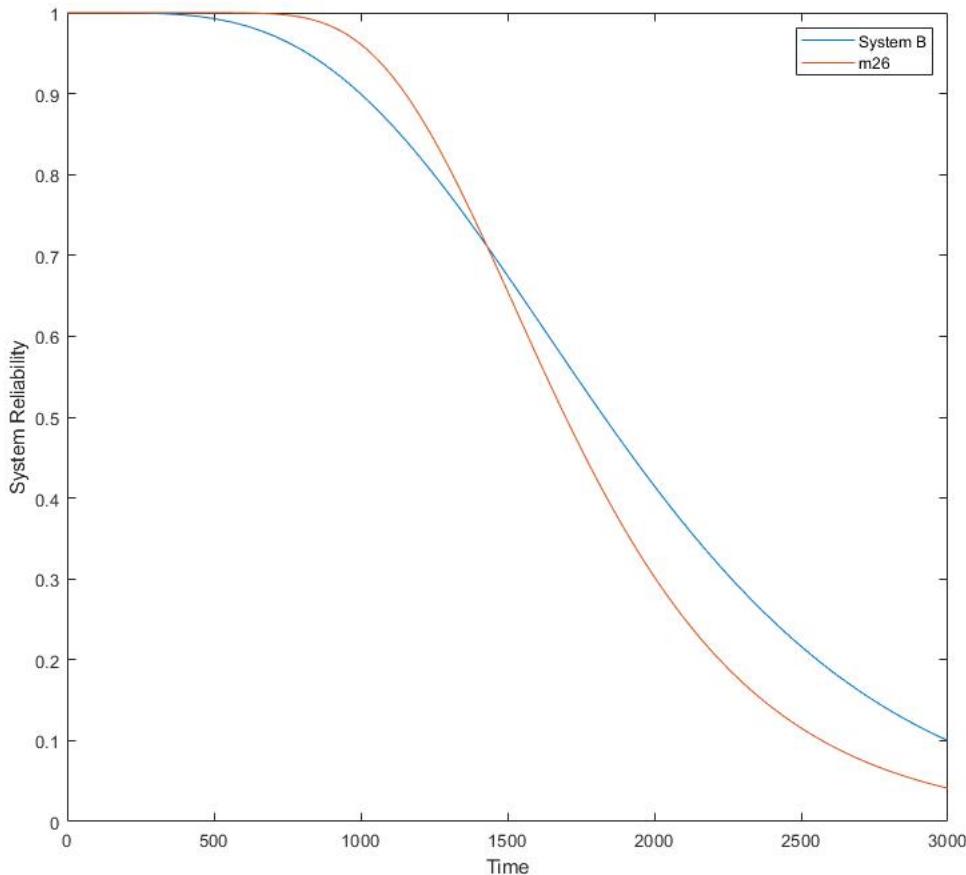


Figura 4.10: Confronto Reliability con sistema A modificato

4.3 Esercizio 3

4.3.1 Traccia

L'architettura di una rete di computer in un sistema bancario è mostrata in figura. L'architettura è chiamata skip-ring network ed è progettata per permettere ai processori di comunicare anche dopo l'avvenimento di un failure in un nodo. Ad esempio, se il nodo 1 fallisce, il nodo 8 può bypassare il nodo fallito instradando i dati sul link alternativo che collega il nodo 8 con il 2. Assumendo

che i link siano perfetti e i nodi abbiano ognuno una reliability R_m , derivare l'espressione per la reliability della rete. Se R_m segue la legge di fallimento esponenziale e il failure rate di ogni nodo è di 0.005 failure all'ora, determinare la reliability del sistema alla fine di un periodo di 48 ore.

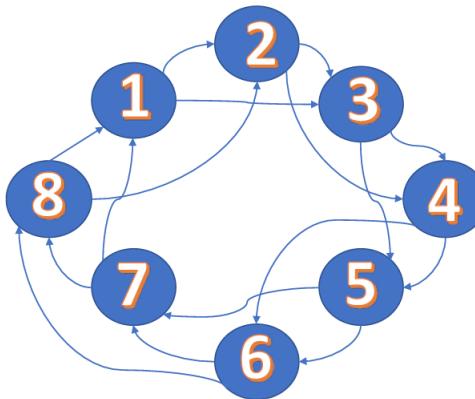


Figura 4.11: Skip-ring network

4.3.2 Soluzione

Dalla traccia dell'esercizio si capisce subito che il sistema fallisce quando falliscono due nodi adiacenti. Il massimo numero di nodi che può quindi fallire è 4 (non consecutivi). Per calcolare la Reliability del sistema si può usare la formula M-out-of-N ($M = 4$ e $N = 8$).

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

Tuttavia essa non tiene in considerazione l'adiacenza dei nodi falliti. Bisogna sottrarre ai vari risultati del coefficiente binomiale della sommatoria le configurazioni di nodi consecutivi (adj_i). La formula modificata del M-out-of-N diventa dunque:

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

Si considerano i casi che vanno da 0 a 4:

- Se $i = 0$ il coefficiente binomiale resta invariato;
- Se $i = 1$ il coefficiente binomiale resta invariato;
- Se $i = 2$ il coefficiente binomiale non deve ammettere le configurazioni con due nodi adiacenti, ossia 8;
- Se $i = 3$ il coefficiente binomiale non deve ammettere le configurazioni con tre nodi adiacenti più quelle in cui due sono adiacenti e non il terzo quindi in totale $8 + 8*4 = 40$;
- Se $i = 4$ il coefficiente binomiale può ammettere solo due casi in cui ci sono 4 nodi funzionanti, di conseguenza verranno escluse 68 configurazioni;.

Avendo definito il vettore $adj_i = [0 \ 0 \ 8 \ 40 \ 68]$, si può calcolare la Reliability del sistema:

$$R_{sys} = \sum_{i=0}^4 \left(\binom{8}{i} - adj_i \right) R_m^{8-i} (1 - R_m)^i = -R_m^8 + 8R_m^7 - 16R_m^6 + 8R_m^5 + 2R_m^4$$

Conoscendo la Reliability dell'intero sistema e sapendo che ogni componente fallisce secondo una distribuzione esponenziale con $\lambda = 0.005$ all'ora, possiamo calcolare la Reliability su un periodo di 48 ore:

$$R_{sys}(t) = -e^{-8\lambda t} + 8e^{-7\lambda t} - 16e^{-6\lambda t} + 8e^{-5\lambda t} + 2e^{-4\lambda t}$$

$$R_{sys}(48) = -e^{-1.92} + 8e^{-1.68} - 16e^{-1.44} + 8e^{-1.2} + 2e^{-0.96} = 0.7288$$

4.3.2.1 Fault Tree

Si può vedere un modo alternativo per calcolare la Reliability del sistema, ossia costruire un **Fault Tree** a partire dalla condizione minima di fallimento, ossia quella in cui due nodi consecutivi falliscono.

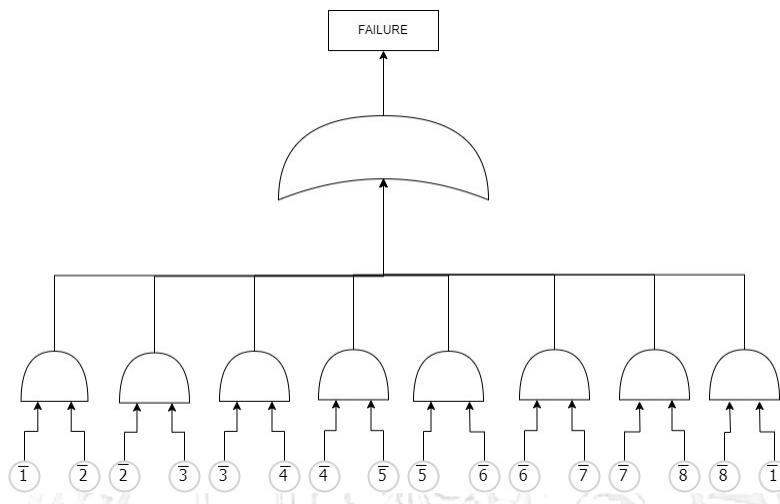


Figura 4.12: Fault Tree

A questo punto è possibile ricavare l'RBD dal Fault Tree e calcolare la reliability. Si ricordi che una porta AND e una porta OR corrispondono rispettivamente a un parallelo e una serie.

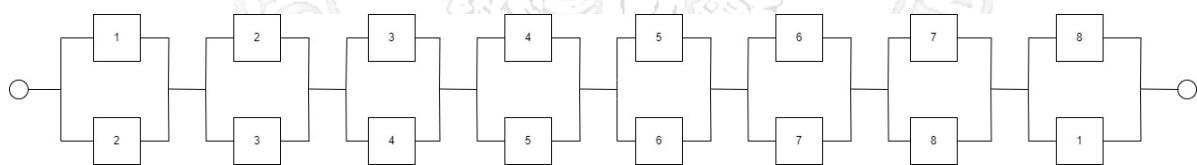


Figura 4.13: RBD

Dunque, la reliability del sistema sarà:

$$R_{sys} = (1 - (1 - R_m)^2)^8$$

Calcolando, come prima, la Reliability su un periodo di 48 ore, si avrà:

$$R_{sys} = 0.6888$$

4.4 Esercizio 4

4.4.1 Traccia

Confronta la Reliability dei seguenti schemi, assumendo un fallimento esponenziale con i seguenti valori:

- $MTTF_A = 900$ ore
- $MTTF_B = 7000$ ore
- $MTTF_C = 1000$ ore

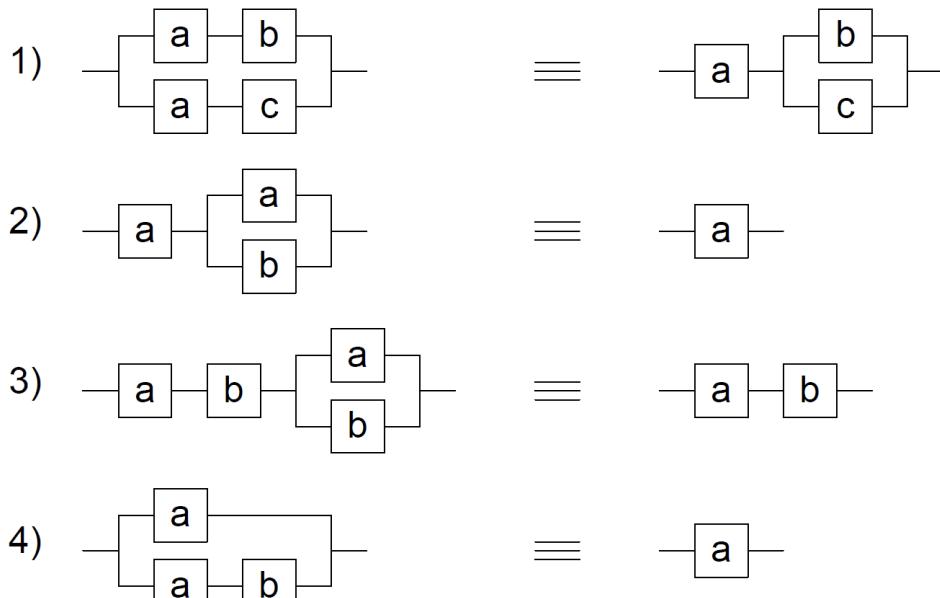


Figura 4.14: RBD da confrontare

4.4.2 Soluzione Punto 1

Si procede calcolando la reliability dei due sistemi tramite lo studio delle serie e dei paralleli.



Figura 4.15: Punto 1

$$R_{sysA} = (1 - (1 - R_a R_b)(1 - R_a R_c))$$

$$R_{sysB} = R_a(1 - (1 - R_b)(1 - R_c))$$

Considerando i seguenti parametri $R_a = e^{-\frac{t}{900}}$ e $R_b = e^{-\frac{t}{7000}}$ e $R_c = e^{-\frac{t}{1000}}$, andiamo a plottare i due sistemi:

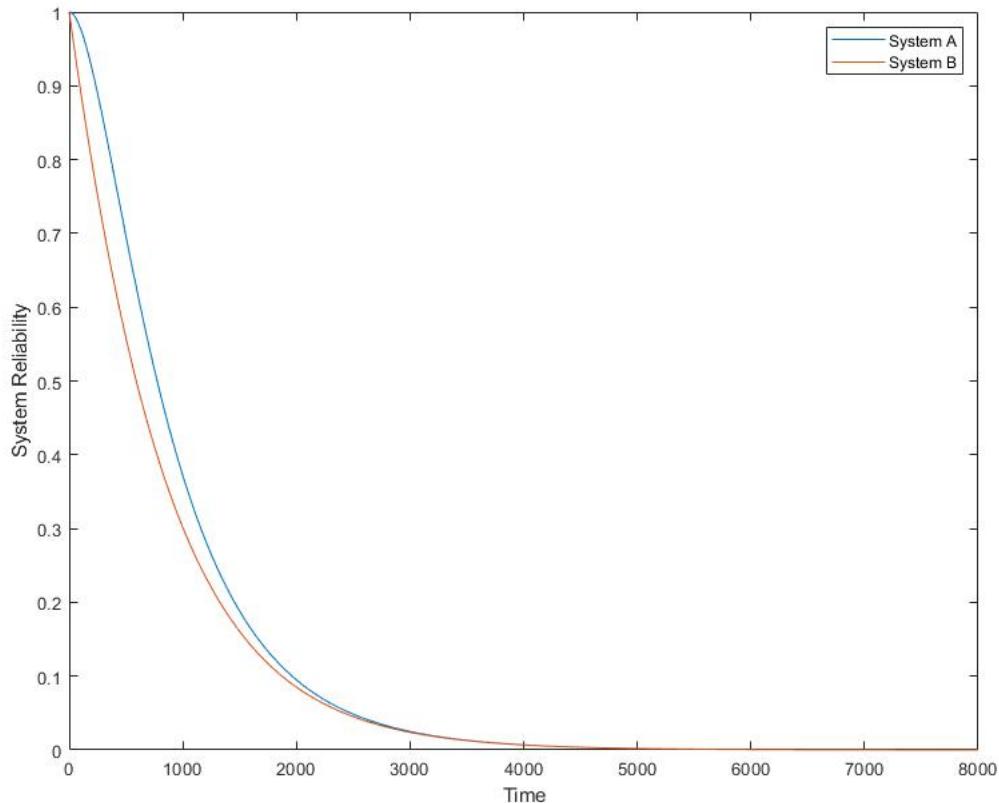


Figura 4.16: Confronto sistemi punto 1

Come si può notare dal grafico ottenuto in Matlab, la reliability del sys_A è maggiore della reliability del sys_B in quanto il fallimento della serie $(a+b)$ oppure della serie $(a+c)$ non implica il fallimento dell'intero sistema, mentre per quanto riguarda il sys_B , il fallimento di a implica il fallimento dell'intero sistema.

4.4.3 Soluzione Punto 2

Si procede calcolando la reliability dei due sistemi tramite lo studio delle serie e dei paralleli.

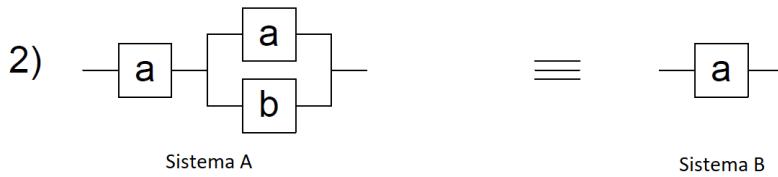


Figura 4.17: Punto 2

$$R_{sysA} = R_a(1 - (1 - R_a)(1 - R_b))$$

$$R_{sysB} = R_a$$

Considerando i seguenti parametri $R_a = e^{-\frac{t}{900}}$ e $R_b = e^{-\frac{t}{7000}}$, andiamo a plottare i due sistemi:

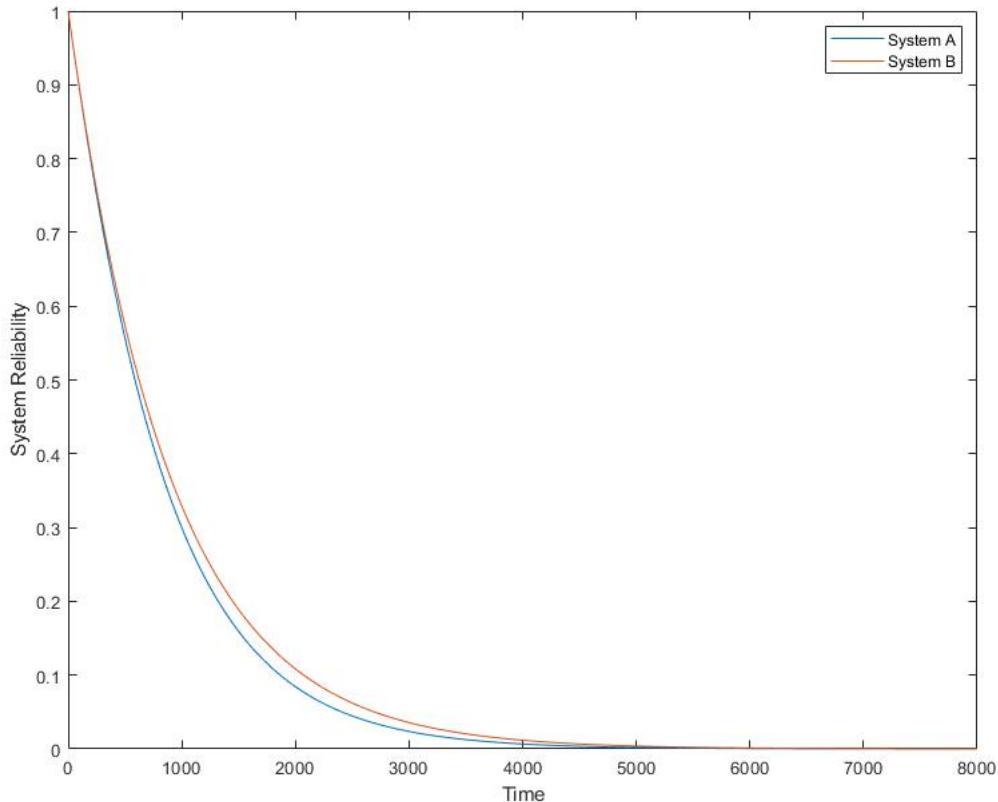


Figura 4.18: Confronto sistemi punto 2

Come si può notare dal grafico ottenuto in Matlab, la reliability del sys_B è maggiore della reliability del sys_A poiché, anche se il numero di success path di sys_A è maggiore, introduce una serie, la quale abbatte la reliability. Dunque conviene utilizzare il sys_B , formato da un singolo componente.

4.4.4 Soluzione Punto 3

Si procede calcolando la reliability dei due sistemi tramite lo studio delle serie e dei paralleli.

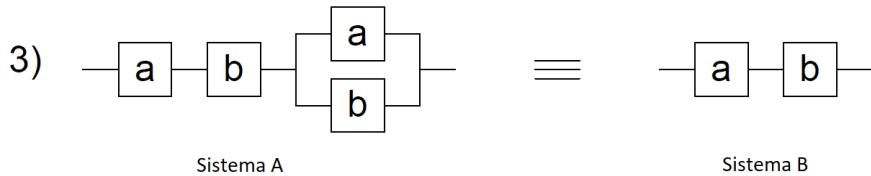


Figura 4.19: Punto 3

$$R_{sysA} = R_a R_b (1 - (1 - R_a)(1 - R_b))$$

$$R_{sysB} = R_a R_b$$

Considerando i seguenti parametri $R_a = e^{-\frac{t}{900}}$ e $R_b = e^{-\frac{t}{7000}}$, andiamo a plottare i due sistemi:

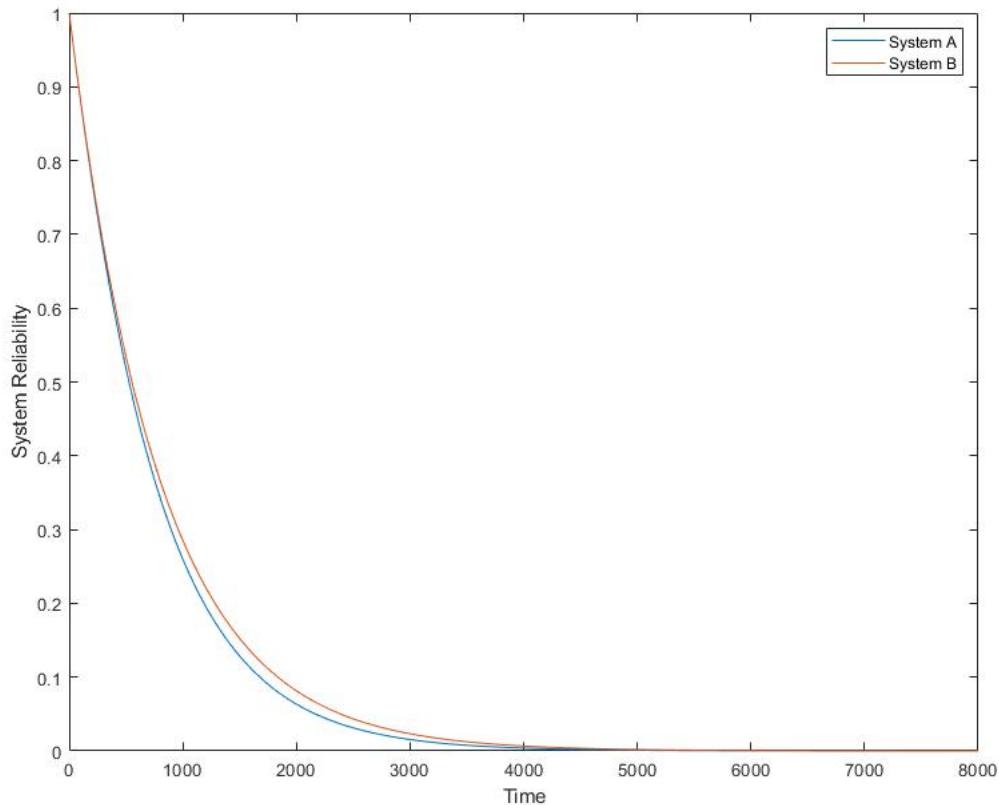


Figura 4.20: Confronto sistemi punto 3

Come si può notare dal grafico ottenuto in Matlab, la reliability del sys_B è maggiore della reliability del sys_A poiché, come detto in precedenza, la serie in più in sys_A diminuisce la reliability totale.

4.4.5 Soluzione Punto 4

Si procede calcolando la reliability dei due sistemi tramite lo studio delle serie e dei paralleli.

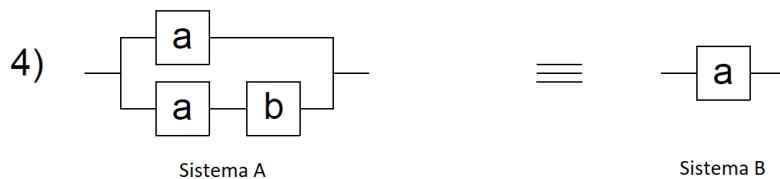


Figura 4.21: Punto 4

$$R_{sysA} = 1 - (1 - R_a)(1 - R_a R_b)$$

$$R_{sysB} = R_a$$

Considerando i seguenti parametri $R_a = e^{-\frac{t}{900}}$ e $R_b = e^{-\frac{t}{7000}}$, andiamo a plottare i due sistemi:

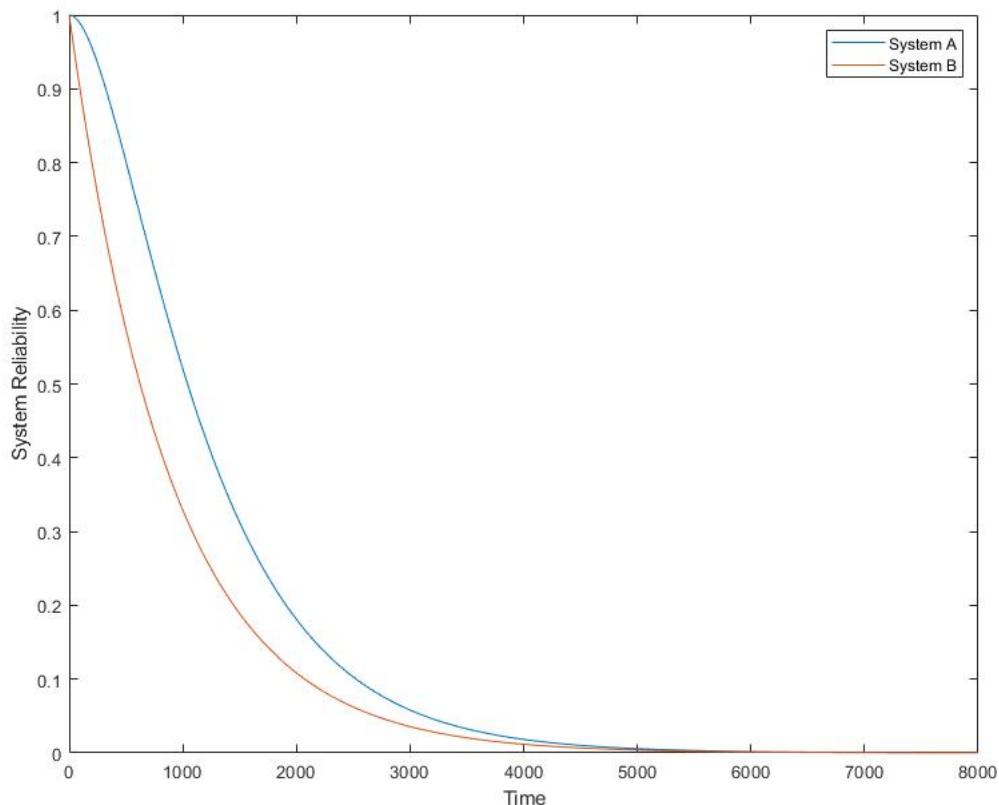


Figura 4.22: Confronto sistemi punto 4

Come si può notare dal grafico ottenuto in Matlab, la reliability del sys_A è maggiore della reliability del sys_B poiché sys_A introduce un parallelo e quindi aggiunge un success path in più rispetto a sys_B .

4.5 Esercizio 5

4.5.1 Traccia

Il sistema mostrato in figura è un sistema di elaborazione per un elicottero. Il sistema ha una ridondanza duale sia per i processori che per le unità di interfacciamento. Vengono utilizzati due bus e ogni bus è replicato. La parte interessante del sistema è l'equipaggiamento di navigazione. Il velivolo può essere completamente guidato usato l'Inertial Navigation System (INS). Se tale INS fallisce, il velivolo può essere guidato attraverso una combinazione di Doppler e AHRS (Altitude Heading Reference System). Di quest'ultima unità, ne sono presenti 3 delle quali una sola è necessaria. I dati dal Doppler e un AHRS possono essere usati in sostituzione del componente INS se esso fallisce. A causa di altri sensori e strumentazioni, sono richiesti entrambi i bus affinché il sistema funzioni in maniera appropriata, indipendentemente dal sistema di navigazione utilizzato.

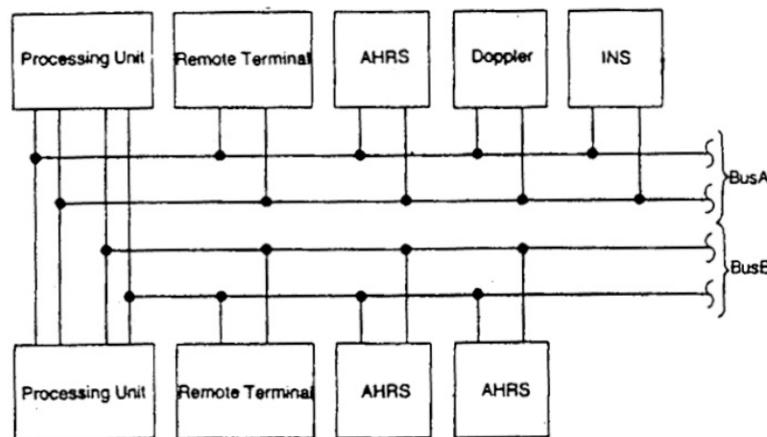


Figura 4.23: Architettura velivolo

1. Disegnare l'RBD del sistema;
2. Disegnare il Fault Tree del sistema ed analizzare i minimal cutset;
3. Calcolare la reliability per un'ora di volo utilizzando i valori di MTTF in tabella. Assumere che venga applicata la legge di fallimento esponenziale e che la fault coverage sia perfetta;

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

Figura 4.24: MTTF dei componenti

4. Ripetere il punto precedente, ma stavolta incorporare un fattore di coverage per la fault detection e riconfigurazione delle unità di elaborazione. Usando gli stessi dati di fallimento, determinare il valore approssimativo di fault coverage richiesto per ottenere (alla fine dell'ora) una reliability di 0.99999.

4.5.2 Soluzione

4.5.2.1 Punto 1

Dallo studio della traccia si evincono alcuni aspetti importanti per la costruzione dell'RBD:

- Il sistema presenta due processori ridondanti, dunque si può costruire un parallelo tra i due componenti;
- Il sistema presenta due unità di interfacciamento, dunque si può costruire un parallelo tra i due componenti;
- Ogni bus è replicato, dunque si possono costruire due paralleli per entrambi i bus;
- I blocchi del sistema di navigazione si possono costruire come un parallelo tra INS e la serie tra il Doppler e il parallelo degli AHRS (solo uno è necessario).

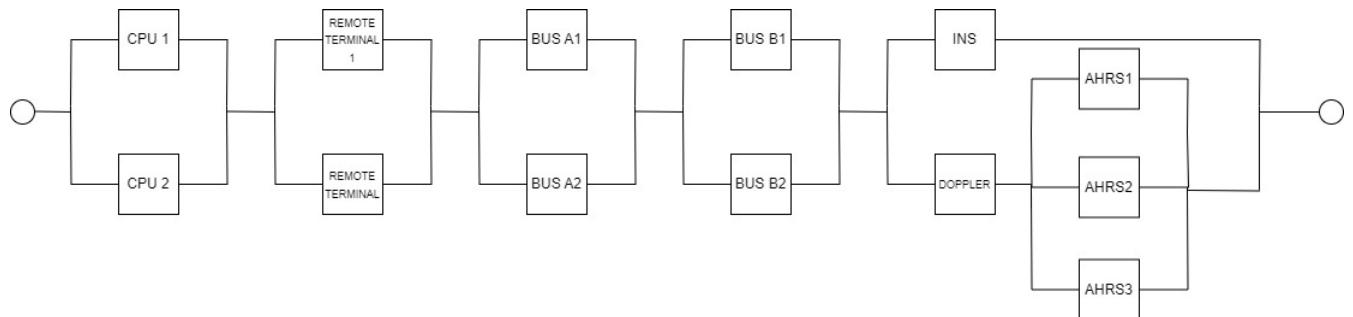


Figura 4.25: RBD elicottero

4.5.2.2 Punto 2

Dallo schema precedente possiamo ricavare il Fault Tree seguendo le seguenti regole :

- i componenti in parallelo nell'RBD sono AND nel Fault Tree;
- i componenti in serie nell'RBD sono OR nel Fault Tree.

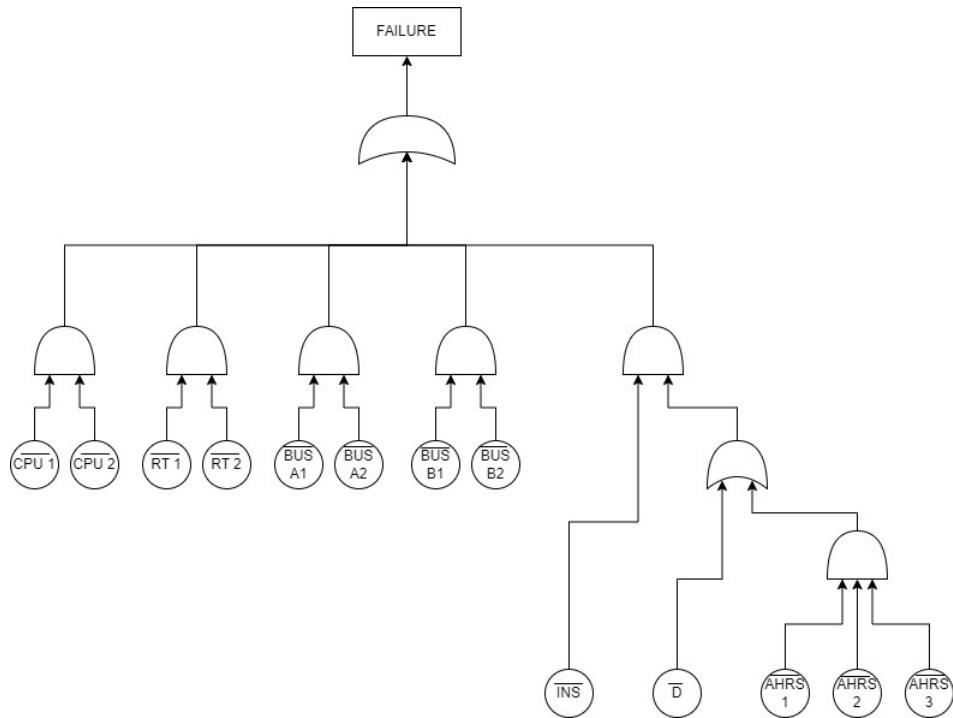


Figura 4.26: Fault Tree elicottero

Esso ci permette di calcolare i minimal cut-set, ossia il numero minimo di componenti che devono fallire affinché fallisca l'intero sistema. Scriviamo quindi il fault tree in forma analitica come somma di prodotti:

$$F = (CPU_1 * CPU_2) + (RT_1 * RT_2) + (A_1 * A_2) + (B_1 * B_2) + \{INS * [D + (AHRS_1 * AHRS_2 * AHRS_3)]\}$$

Quindi i minimal cut-set (somme di prodotti) sono:

- Il fallimento di entrambi i bus A;
- Il fallimento di entrambi i bus B;
- Il fallimento di entrambi i processori;
- Il fallimento di entrambi i terminali;
- Il fallimento di INS insieme al fallimento del Doppler, oppure il fallimento dell'INS insieme al fallimento di tutti gli AHRS.

4.5.2.3 Punto 3

A questo punto possiamo calcolare la Reliability dell'intero sistema utilizzando l'RBD del punto 1:

$$R_{sys} = (1 - (1 - R_{CPU})^2)(1 - (1 - R_{RT})^2)(1 - (1 - R_{BUSA})^2)(1 - (1 - R_{BUSB})^2)$$

$$(1 - (1 - R_{INS})(1 - (R_D(1 - (1 - R_{AHRS})^3)))$$

Le varie Reliability possono essere sostituite seguendo l'ipotesi della legge di fallimento esponenziale:

- $R_{CPU} = e^{-\frac{t}{10000}}$
- $R_{RT} = e^{-\frac{t}{4500}}$
- $R_{AHRS} = e^{-\frac{t}{2000}}$
- $R_{INS} = e^{-\frac{t}{2000}}$
- $R_D = e^{-\frac{t}{500}}$
- $R_{BUSA} = R_{BUSB} = e^{-\frac{t}{60000}}$

Grazie al software MATLAB si può vedere il grafico della Reliability rispetto al tempo (in ore).

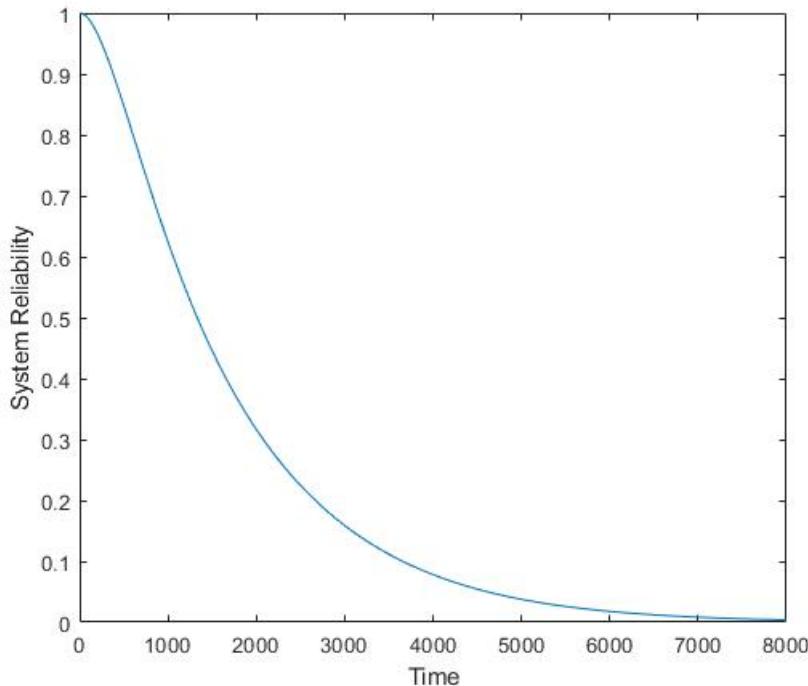


Figura 4.27: Reliability rispetto al tempo

Andando a impostare $t = 1$ ora si può trovare il valore preciso di Reliability dopo un'ora di funzionamento:

$$R_{sys}(1h) = 0.999998941322750$$

4.5.2.4 Punto 4

Per calcolare il fattore di coverage bisogna modificare l'espressione del parallelo tra le due CPU, poiché in questo caso non si ha più una coverage perfetta come nel caso precedente. Quindi:

$$R_{CPU1} + c(1 - R_{CPU1})R_{CPU2}$$

Si sta sommando la Reliability di CPU_1 con la Reliability di CPU_2 nel caso in cui CPU_1 fallisca ma il sistema ridireziona il flusso su CPU_2 per nascondere il fallimento.

Infine possiamo ricavare il fattore c risolvendo la seguente equazione:

$$(R_{CPU1} + c(1 - R_{CPU1})R_{CPU2})(1 - (1 - R_{RT})^2)(1 - (1 - R_{BUSA})^2)(1 - (1 - R_{BUSB})^2)$$

$$(1 - (1 - R_{INS})(1 - (R_D(1 - (1 - R_{AHRs})^3))) = 0.99999$$

$$c = 0.910573$$

Dunque il sistema di fault detection rileverà il guasto ridirezionando il flusso verso la seconda unità di elaborazione con una probabilità di circa il 91%.



Capitolo 5

FFDA

5.1 Traccia

Eseguire l'analisi per MercuryErrorLog.txt e BGLError-Log.txt:

- grafico del conteggio delle tuple + CWIN selezionato;
- raggruppare le voci con il CWIN selezionato;
- distribuzioni di affidabilità empirica;
- adattamento dell'affidabilità empirica (provare tutti i modelli - esponenziale, iperesponeziale, weibull);
- analisi dei risultati della modellizzazione tramite il test KS Confronto dei risultati tra i sistemi.

Condurre l'analisi per rispondere alle seguenti domande:

1. È possibile utilizzare le stesse finestre di coalscenza tra diversi nodi (MERCURY, BG/L) e categorie di errore (MERCURY)?
2. Qual è la relazione tra l'affidabilità del sistema e l'affidabilità dei nodi?
3. Esistono dei colli di bottiglia dell'affidabilità (cioè i principali responsabili del numero totale di guasti)?
4. Nodi funzionali simili (ad esempio due nodi di calcolo Mercury tg-cX o nodi BG/L I/O Ri:Mx:Nz) presentano parametri di affidabilità simili?
5. Esiste una relazione tra i tipi di errore e il nodo (MERCURY)?

5.2 Field Failure Data Analysis (FFDA)

La **Field Failure Data Analysis** è una tecnica di misurazione diretta volta a misurare attributi di *dependability* di un sistema in esecuzione in un ambiente di lavoro reale e su cui viene applicato un workload reale. Tale tecnica consiste nell'osservare gli errori ed i fallimenti che si verificano durante il normale funzionamento del sistema al fine di poter ottenere una serie di informazioni:

- Stima del tempo medio di reliability del sistema;
- Dati per la realizzazione del fitting delle curve di reliability empirica.

A partire dai fallimenti e dagli errori del sistema, la FFDA è composta di quattro macro-fasi:

1. **Logging Collection:** in tale fase sono raccolti e memorizzati i dati relativi al sistema mentre lo stesso è in funzione. Per svolgere tale fase è necessario definire quali dati raccogliere e in che modo. Ciò è in genere definito grazie ad uno studio preliminare del sistema e dell'ambiente di funzionamento, per poter identificare le tecniche di raccolta più adatte e decidere se è conveniente o meno sviluppare un sistema di monitoraggio ad hoc;
2. **Filtering:** durante la fase di filtering sono estratte solo le informazioni di interesse relative a tutti i dati raccolti in precedenza.
3. **Manipulation:** fase in cui i dati sono aggregati per mezzo di un'operazione di coalescenza. Tale operazione permette di raggruppare errori relativi allo stesso guasto all'interno di uno stesso elemento detto **tupla** per ridurre in maniera ulteriore e notevole la dimensione del dataset iniziale.
4. **Analysis:** fase finale in cui si svolge un'analisi statistica per poter caratterizzare i dati estratti e filtrati nelle fasi precedenti.

5.3 Mercury

Il sistema Mercury è un cluster composto da nodi IBM organizzati in tre livelli. Ogni cluster possiede un'architettura three-layer con l'aggiunta di un nodo di management (tg-master).

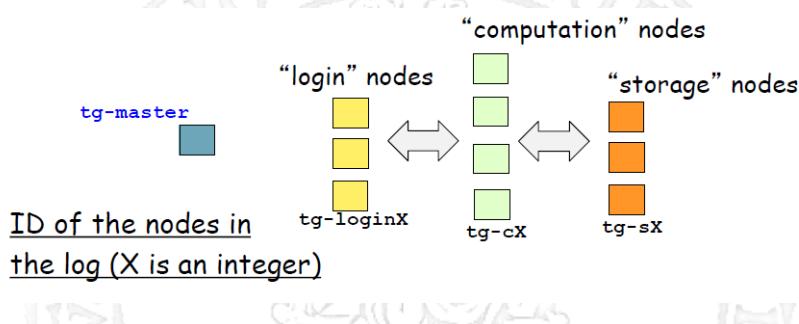


Figura 5.1: Architettura Mercury

Si descrivono in seguito i dettagli tecnici di questo cluster e la metodologia di event recording. L'architettura generica di tali cluster si suddivide in tre strati :

1. nodi di login, utilizzati dagli utenti finali per accedere al cluster, compilare il software e presentare e controllare i jobs;
2. nodi di calcolo, assegnati agli utenti per eseguire i jobs seriali o paralleli;
3. nodi di massa, contenenti lo spazio su disco utilizzato per contenere i datasets per l'esecuzione dei jobs sul cluster.

Tutti sono gestiti da un unico nodo di management: tg-master.

Il sistema in esame è stato monitorato durante un periodo di circa 3 mesi. Gli eventi nel file di Log sono collezionati dal demone syslog, ciascuna riga del file di log contiene:

- **Timestamp** : ora, giorno, mese, anno in cui si è verificato l'evento, con una risoluzione del secondo;
- **Name** : nome/identificativo del nodo che genera il messaggio;
- **Type** : categoria a cui l'errore viene attribuito;
- **Msg** : testo libero che descrive l'evento.

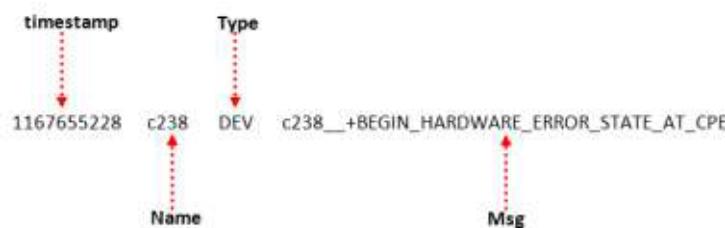


Figura 5.2: Formato messaggio di log Mercury

Sono individuate sei categorie di errore che si elencano in seguito :

1. **Device (DEV)**: errori legati alle periferiche ed alle schede PCI;
2. **Memory (MEM)**: errori non correggibili della memoria;
3. **Network (NET)**: problemi di comunicazione sollevati da una macchina;
4. **Input/Output (I/O)**: problemi differenti: dell'interfaccia SCSI, danneggiamento di settori...;
5. **Processor (PRO)**: eccezioni del processore;
6. **Other (OTH)**: errori non attribuibili ad alcuna delle precedenti categorie.

5.4 Blue Gene/L

Il data center *BlueGene* è un supercomputer IBM a parallelismo massivo, e di dimensioni maggiori del sistema Mercury. La sua architettura è formata dai seguenti componenti:

- **Rack (RX)**: sistema standard d'installazione fisica di componenti hardware a scaffale. Il sistema è caratterizzato da 104 rack, ciascuno dei quali viene scomposto in due midplane (MX);
- **Nodo (NX)**: ciascun midplane è caratterizzato da 16 nodi, dunque in ciascun rack sono presenti 32 nodi; 92 Dependability

- **Compute Card (JX)**: ciascun nodo è caratterizzato da 16 unità di calcolo computazionale, dunque in ciascun rack sono presenti 512 nodi;
- **I/O Card (J18)**: solo in alcuni nodi è presente una card I/O;
- **Compute Chip (U01,U11)**: ciascuna compute card è caratterizzata da 2 compute chip, dotati di due CPU ciascuno.

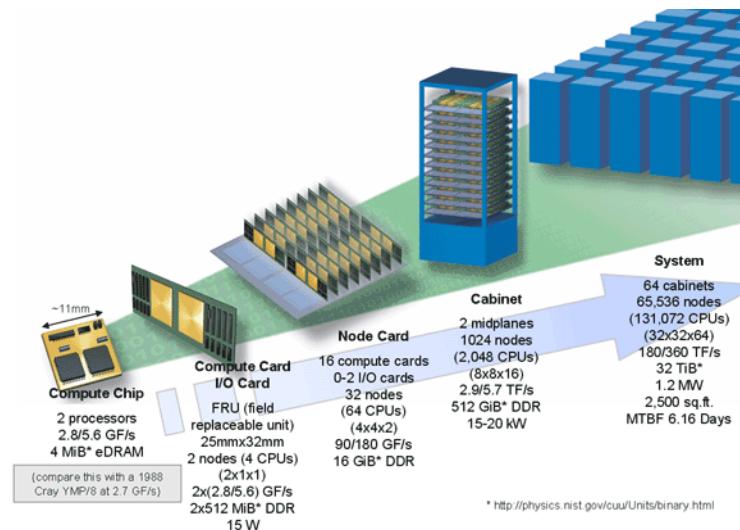


Figura 5.3: Architettura BlueGene

Il file di log di BlueGene è composto dalle seguenti informazioni:

- **Timestamp**: ora, giorno, mese, anno in cui si è verificato l'evento, con una risoluzione del secondo;
- **Originating node**: nodo che ha generato il messaggio;
- **Originating Card**: categoria/card a cui l'errore viene attribuito;
- **Message**: testo che descrive l'evento;

5.5 Data Manipulation: Mercury

Obiettivo della fase di data manipulation è identificare, a partire dal log del sistema, i fallimenti del sistema. Per identificare i fallimenti di un sistema si utilizza la tecnica di *coalescenza*, in grado di rilevare la correlazione tra le entry del file di log. La coalescenza è una tecnica per il filtraggio e per la manipolazione dei dati in grado di ridurre l'insieme delle informazioni raccolte durante la fase di *logging collection*. Tale tecnica, tramite operazioni di aggregazione o di cancellazione, consente l'eliminazione di dati ridondanti o equivalenti tra di loro; ciò è necessario in quanto un guasto genera in un sistema più fallimenti e dato che gli effetti di un guasto si propagano in un sistema, essi vengono rilevati più volte. Per cui si vanno a raggruppare tutti gli errori ed alerts relativi al medesimo guasto in un unico evento.

Nel caso in esame si è optato per un tipo di coalescenza spaziale: si parla di coalescenza spaziale quando entry differenti vengono raggruppate secondo un algoritmo che misura la distanza tra esse a livello temporale. Tale algoritmo è un algoritmo di tupling, che genera delle entry dette **tuple**; esso opera andando a calcolare la differenza tra due timestamp di entry consecutive e confronta tale risultato con il valore di finestra **W**, in particolare le due entry sono fuse tra loro se la differenza dei timestamp è minore di **W**.

```

IF  $t(X_{i+1}) - t(X_i) < W$  THEN
    Add  $X_{i+1}$  to the tuple
    • where:
        •  $X_i$  is the  $i$ -th entry in the log;
        •  $t(X_i)$  is the timestamp of the  $X_i$  entry;
        •  $W$  is a configurable time window.

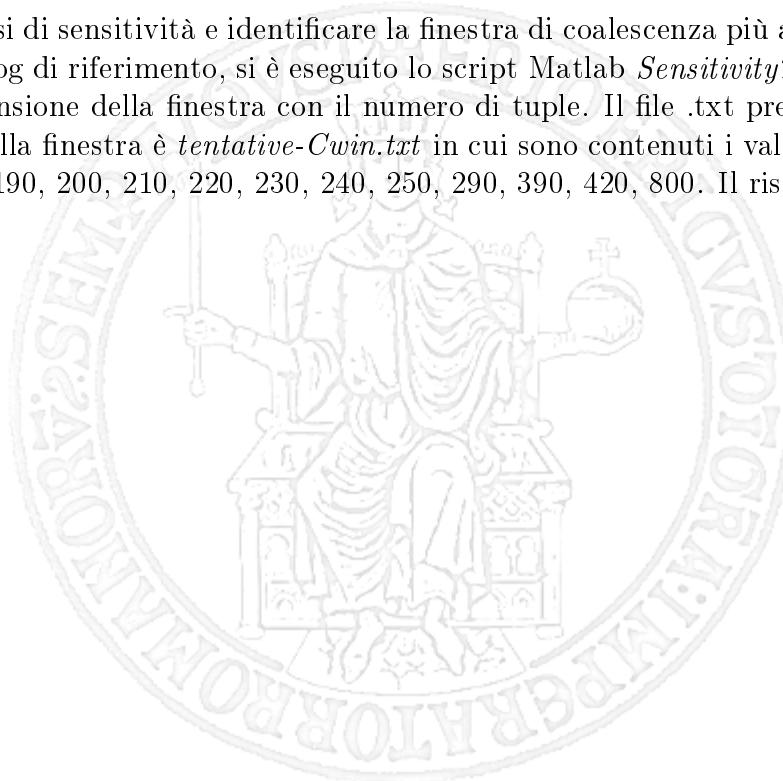
```

Figura 5.4: Pseudo codice coalescenza temporale

Dunque, per poter utilizzare tale tipologia di coalescenza è necessario selezionare una **finestra di coalescenza** che abbia un dimensionamento tale da non influenzare i risultati del tupling ed evitando le problematiche di *collision* e *truncation*.

5.5.1 Sensitivity Analysis: Mercury

Per condurre l'analisi di sensitività e identificare la finestra di coalescenza più adatta al raggruppamento in tuple del log di riferimento, si è eseguito lo script Matlab *SensitivityTuple.m* per mettere in relazione la dimensione della finestra con il numero di tuple. Il file .txt preso come riferimento per le dimensioni della finestra è *tentative-Cwin.txt* in cui sono contenuti i valori numerici: 10, 50, 100, 120, 150, 180, 190, 200, 210, 220, 230, 240, 250, 290, 390, 420, 800. Il risultato è il seguente:



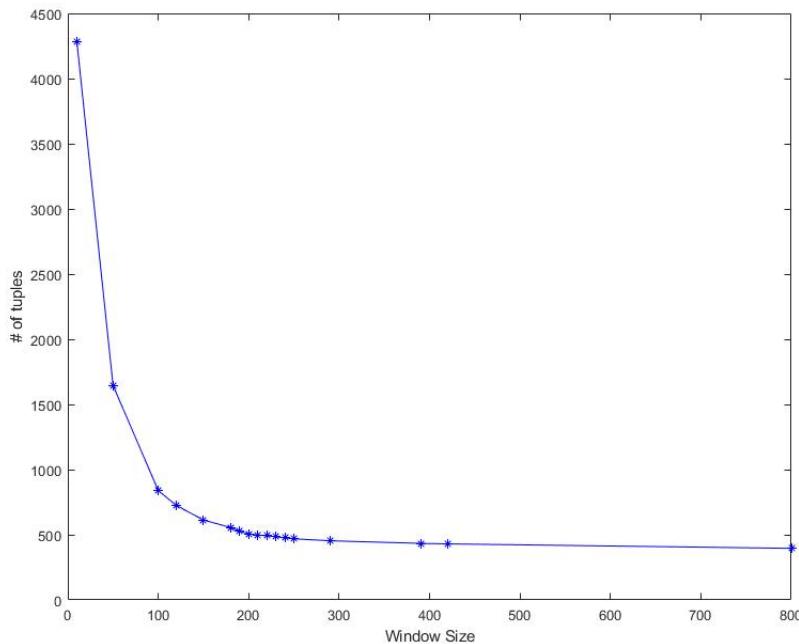


Figura 5.5: Analisi di sensitività - Mercury

Si seleziona, tra tutti i valori di W calcolati, quello che produce il decremento più marcato, ossia il cosiddetto valore di ginocchio, che rappresenta il punto in cui il numero di tuple smette di decrescere rapidamente all'aumentare della finestra di coalescenza. Quindi, nel caso di esame si è scelto un valore di **220 secondi** che presenta un numero di tuple pari a 493.

Successivamente tale valore è stato utilizzato per lo script *tupling_with_CWIN.sh* che analizza il file di log per produrre in output le tuple e file utili per successive analisi. Nel dettaglio:

- **lengths.txt**: indica la durata dei fallimenti e di conseguenza la lunghezza delle singole tuple.
- **interrivals.txt**: sono i tempi di interarrivo tra le tuple (TTF).
- **startingPoints.txt**: timestamp del punto di inizio della tupla.

Grazie al file *interrivals.txt* sarà possibile realizzare un'analisi sulla reliability del sistema.

Prima di andare alla fase successiva si fa un'analisi di troncamenti e collisioni data la finestra di coalescenza scelta. I primi vengono generati da una finestra di coalescenza non sufficientemente grande da raggruppare fallimenti dello stesso tipo, fornendo una stima pessimistica dei guasti. Le collisioni, invece, vengono generate da una finestra di coalescenza troppo grande che può includere guasti dello stesso tipo e di conseguenza fornendo una stima ottimistica dei guasti. Grazie all'aiuto di uno script Python si vanno ad analizzare le tuple che presentano i problemi descritti in precedenza.

Possiamo vedere un esempio di troncamento nella figura successiva:

The screenshot shows two separate terminal windows. The top window is titled 'tuple_29.txt' and displays the following truncated log entry:

```
1167883322 tg-c735 DEV +BEGIN HARDWARE ERROR STATE AT CPE
1167883322 tg-c735 DEV +Platform PCI Component Error Info Section
1167883322 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
```

The bottom window is titled 'tuple_30.txt' and displays a similar truncated log entry:

```
1167894347 tg-c735 DEV +BEGIN HARDWARE ERROR STATE AT CPE
1167894347 tg-c735 DEV +Platform PCI Component Error Info Section
1167894347 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
```

Figura 5.6: Troncamento tuple 29-30

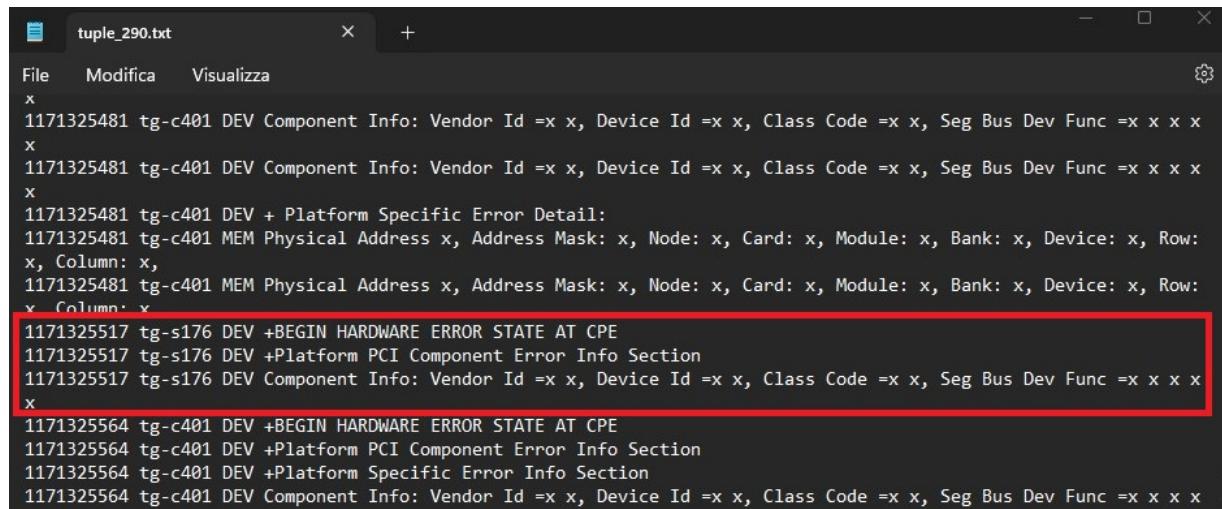
Come si può vedere, il guasto appartiene allo stesso nodo e alla stessa categoria e di conseguenza il raggruppamento è fallito a causa della grandezza della finestra.

Per quanto riguarda le collisioni si possono fare diverse considerazioni:

The screenshot shows a single terminal window titled 'tuple_290.txt'. It displays a log entry followed by several entries from another node. A red box highlights a specific sequence of entries from the second node:

```
1171321357 tg-c401 DEV +BEGIN HARDWARE ERROR STATE AT CPE
1171321357 tg-c401 DEV +Platform PCI Component Error Info Section
1171321357 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
1171321373 tg-c401 DEV +BEGIN HARDWARE ERROR STATE AT CPE
1171321373 tg-c401 DEV +Platform PCI Component Error Info Section
1171321373 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
1171321410 tg-c401 DEV +BEGIN HARDWARE ERROR STATE AT CPE
1171321429 tg-c027 PRO +BEGIN HARDWARE ERROR STATE AT CMC
1171321429 tg-c027 PRO +BEGIN HARDWARE ERROR STATE AT CMC
1171321429 tg-c027 PRO +BEGIN HARDWARE ERROR STATE AT CMC
1171321429 tg-c027 PRO +END HARDWARE ERROR STATE AT CMC
1171321429 tg-c027 PRO Device Error Info Section
1171321429 tg-c027 PRO Device Error Info Section
1171321429 tg-c027 PRO Error Map: x
1171321429 tg-c027 PRO Error Map: x
1171321429 tg-c027 PRO Error Map: x
1171321474 tg-c401 DEV +BEGIN HARDWARE ERROR STATE AT CPE
1171321474 tg-c401 DEV +Platform PCI Component Error Info Section
1171321474 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
1171321494 tg-c401 DEV +BEGIN HARDWARE ERROR STATE AT CPE
1171321494 tg-c401 DEV +Platform PCI Component Error Info Section
1171321494 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 5.7: Collisione 1 tupla 290

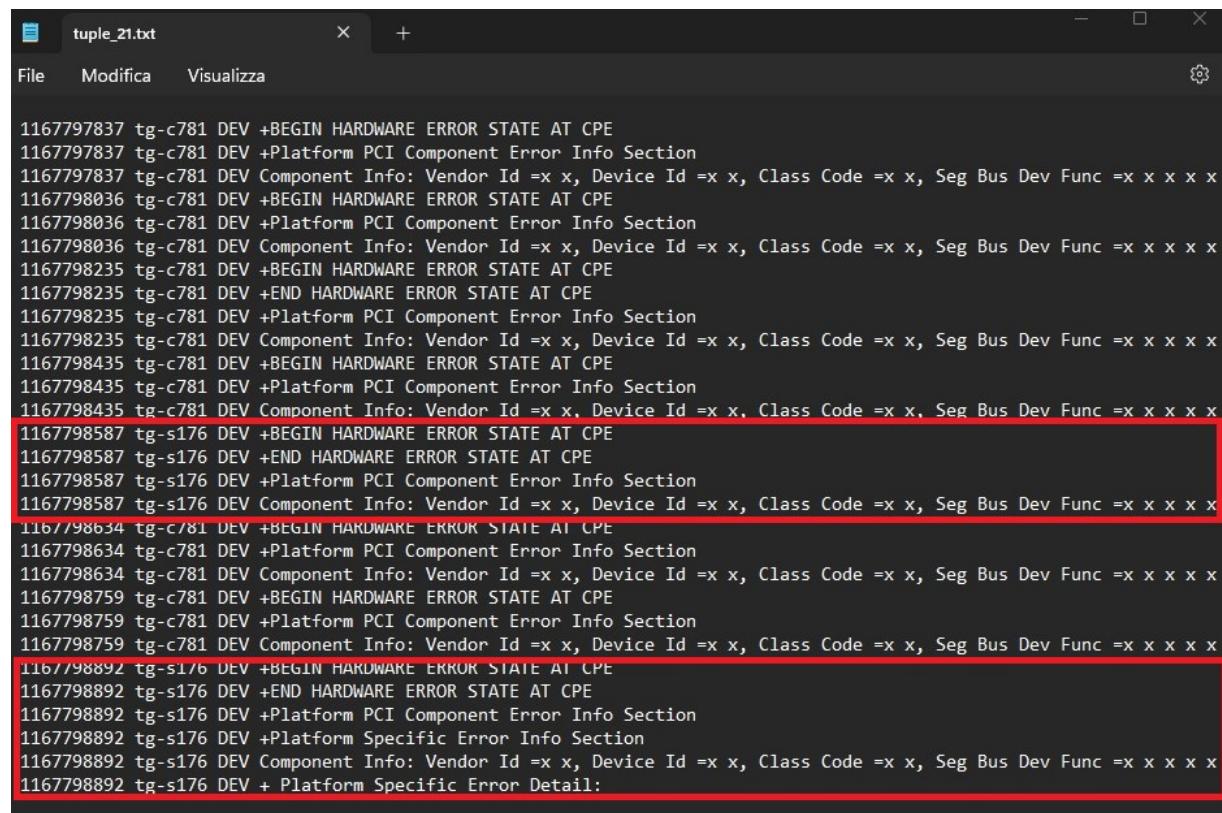


```
tuple_290.txt
File Modifica Visualizza
x
1171325481 tg-c401 DEV Component Info: Vendor Id =x x, Device Id =x x, Class Code =x x, Seg Bus Dev Func =x x x x
x
1171325481 tg-c401 DEV Component Info: Vendor Id =x x, Device Id =x x, Class Code =x x, Seg Bus Dev Func =x x x x
x
1171325481 tg-c401 DEV + Platform Specific Error Detail:
1171325481 tg-c401 MEM Physical Address x, Address Mask: x, Node: x, Card: x, Module: x, Bank: x, Device: x, Row: x, Column: x,
1171325481 tg-c401 MEM Physical Address x, Address Mask: x, Node: x, Card: x, Module: x, Bank: x, Device: x, Row: x, Column: x
1171325517 tg-s176 DEV +BEGIN HARDWARE ERROR STATE AT CPE
1171325517 tg-s176 DEV +Platform PCI Component Error Info Section
1171325517 tg-s176 DEV Component Info: Vendor Id =x x, Device Id =x x, Class Code =x x, Seg Bus Dev Func =x x x x
x
1171325564 tg-c401 DEV +BEGIN HARDWARE ERROR STATE AT CPE
1171325564 tg-c401 DEV +Platform PCI Component Error Info Section
1171325564 tg-c401 DEV +Platform Specific Error Info Section
1171325564 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
```

Figura 5.8: Collisione 2 tupla 290

Si può dire in questo caso che la causa delle collisioni sia dovuta a una questione di casualità poiché, come si evince in altre tuple, il nodo tg-c401 fallisce spesso, di conseguenza può capitare che altri guasti possano nascere durante il suo lungo intertempo.

Di contro, si può vedere nella figura successiva una collisione causata dalla finestra di coalescenza:



```
tuple_21.txt
File Modifica Visualizza
x
1167797837 tg-c781 DEV +BEGIN HARDWARE ERROR STATE AT CPE
1167797837 tg-c781 DEV +Platform PCI Component Error Info Section
1167797837 tg-c781 DEV Component Info: Vendor Id =x x, Device Id =x x, Class Code =x x, Seg Bus Dev Func =x x x x x
1167798036 tg-c781 DEV +BEGIN HARDWARE ERROR STATE AT CPE
1167798036 tg-c781 DEV +Platform PCI Component Error Info Section
1167798036 tg-c781 DEV Component Info: Vendor Id =x x, Device Id =x x, Class Code =x x, Seg Bus Dev Func =x x x x x
1167798235 tg-c781 DEV +BEGIN HARDWARE ERROR STATE AT CPE
1167798235 tg-c781 DEV +END HARDWARE ERROR STATE AT CPE
1167798235 tg-c781 DEV +Platform PCI Component Error Info Section
1167798235 tg-c781 DEV Component Info: Vendor Id =x x, Device Id =x x, Class Code =x x, Seg Bus Dev Func =x x x x x
1167798435 tg-c781 DEV +BEGIN HARDWARE ERROR STATE AT CPE
1167798435 tg-c781 DEV +Platform PCI Component Error Info Section
1167798435 tg-c781 DEV Component Info: Vendor Id =x x, Device Id =x x, Class Code =x x, Seg Bus Dev Func =x x x x x
1167798587 tg-s176 DEV +BEGIN HARDWARE ERROR STATE AT CPE
1167798587 tg-s176 DEV +END HARDWARE ERROR STATE AT CPE
1167798587 tg-s176 DEV +Platform PCI Component Error Info Section
1167798587 tg-s176 DEV Component Info: Vendor Id =x x, Device Id =x x, Class Code =x x, Seg Bus Dev Func =x x x x x
1167798634 tg-c781 DEV +BEGIN HARDWARE ERROR STATE AT CPE
1167798634 tg-c781 DEV +Platform PCI Component Error Info Section
1167798634 tg-c781 DEV Component Info: Vendor Id =x x, Device Id =x x, Class Code =x x, Seg Bus Dev Func =x x x x x
1167798759 tg-c781 DEV +BEGIN HARDWARE ERROR STATE AT CPE
1167798759 tg-c781 DEV +Platform PCI Component Error Info Section
1167798759 tg-c781 DEV Component Info: Vendor Id =x x, Device Id =x x, Class Code =x x, Seg Bus Dev Func =x x x x x
1167798892 tg-s176 DEV +BEGIN HARDWARE ERROR STATE AT CPE
1167798892 tg-s176 DEV +END HARDWARE ERROR STATE AT CPE
1167798892 tg-s176 DEV +Platform PCI Component Error Info Section
1167798892 tg-s176 DEV +Platform Specific Error Info Section
1167798892 tg-s176 DEV Component Info: Vendor Id =x x, Device Id =x x, Class Code =x x, Seg Bus Dev Func =x x x x x
1167798892 tg-s176 DEV + Platform Specific Error Detail:
```

Figura 5.9: Collisione tupla 21

5.6 Data Analysis: Mercury

In merito allo studio relativo alla reliability del sistema, è possibile partire dai risultati ottenuti in fase di data manipulation, in modo da estrarre la distribuzione di probabilità del **time to failure** (TTF) del sistema. Di conseguenza si va a costruire la reliability empirical del sistema come $1 - TTF$.

Ricordiamo che la *reliability* indica la probabilità che sistema funzioni per un tempo t dall'ultimo fallimento e non fallisca prima di tale tempo t . Viceversa, la *unreliability* indica la probabilità che il sistema fallisca dopo un tempo t dall'ultimo fallimento.

Viene mostrato il grafico delle curve del TTF empirico e reliability empirica:

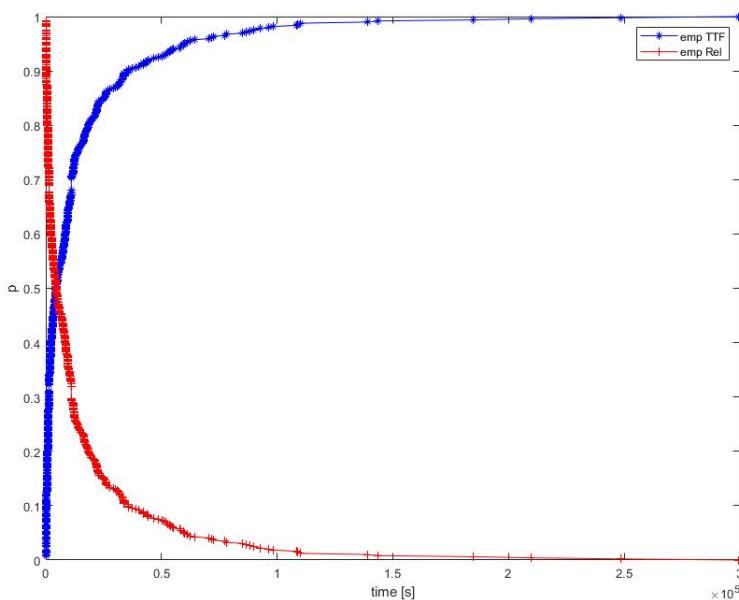


Figura 5.10: TTF e Reliability Empiriche - Mercury

5.6.1 Studio della Reliability

Bisogna cercare una distribuzione teorica nota per rappresentare il più possibile la Reliability empirica trovata precedentemente. A seconda della distribuzione possiamo dedurre la tipologia dei fallimenti presenti nei dati. Utilizzando il comando Matlab *cftool* per il fitting, sono state applicate le seguenti distribuzioni:

- **Esponenziale:** $y = a * e^{-bt}$;
- **Iperesponenziale:** $y = a * e^{bt} + c * e^{dt}$;
- **Weibull:** $y = e^{-(l*t)^a}$

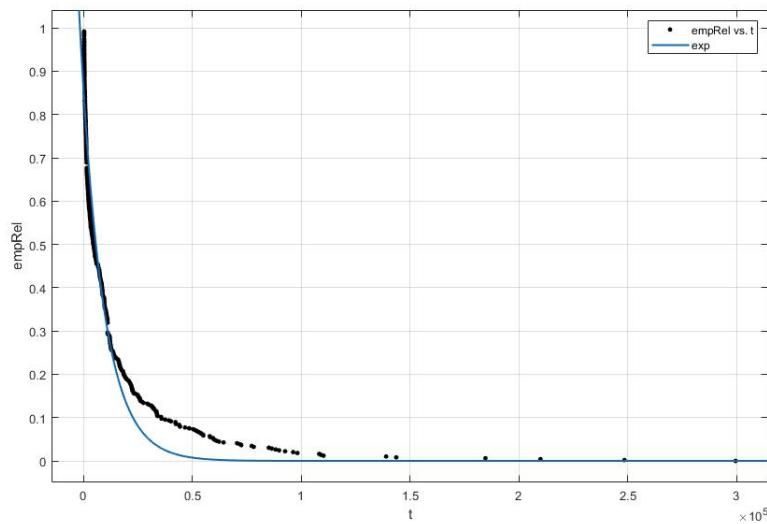


Figura 5.11: Fitting esponenziale - Mercury

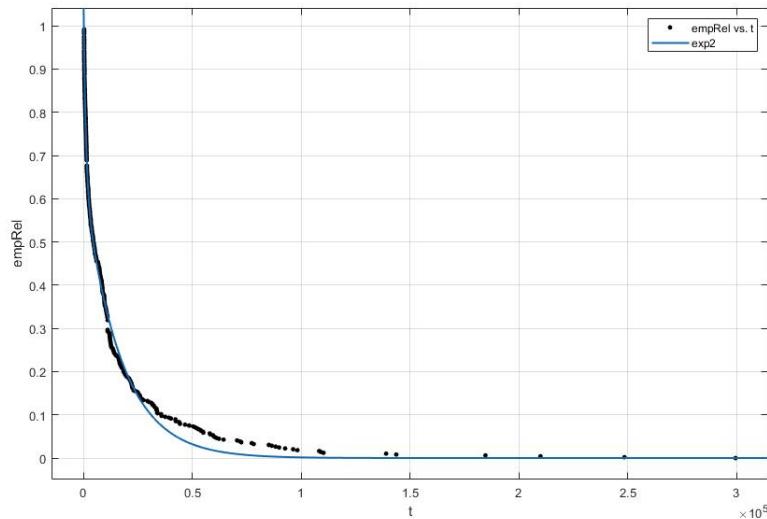


Figura 5.12: Fitting iperesponenziale - Mercury

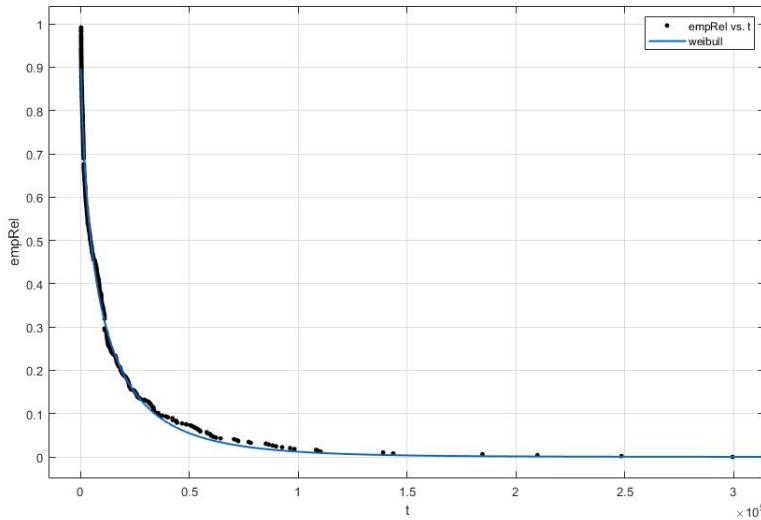


Figura 5.13: Fitting Weibull - Mercury

5.6.2 Goodness of Fitting

Per scegliere la migliore distribuzione notevole è possibile confrontare i valori di due parametri fondamentali:

- **SSE (Sum of Square Errors)**: somma quadratica degli errori; per avere una buona approssimazione deve essere vicina allo zero.
- **R-SQUARE**: misura il legame tra la variabilità dei dati e la correttezza del modello statistico utilizzato; deve essere il più possibile vicina a 1.

I valori ottenuti tramite *cftool* di Matlab sono i seguenti:

Table of Fits							
Fit name	Data	Fit type	SSE	R-square	D.F.E.	Adj R-sq	RMSE
exp	empRel vs. t	exp1	1.5965	0.9574	455	0.9573	0.0592
exp2	empRel vs. t	exp2	0.1328	0.9965	453	0.9964	0.0171
weibull	empRel vs. t	exp(-(b*x)^c)	0.3061	0.9918	455	0.9918	0.0259

Figura 5.14: Table of Fits - Mercury

Come si può notare dalla tabella, la iperesponenziale presenta i parametri migliori di tutte e tre le distribuzioni.

5.6.2.1 Test di Kolmogorov-Smirnov

I parametri studiati in precedenza indicano già una corretta approssimazione dei modelli, tuttavia si esegue il **test non parametrico di Kolmogorov-Smirnov** per avere conferma della bontà del fitting. Per eseguire il test è stato utilizzato il comando Matlab *kstest2* con in input il vettore empirico e il vettore stimato. In output si ottengono i seguenti parametri:

- **H**: indica se il test di ipotesi nulla è rigettato (se $H=0 \Rightarrow$ accetta l'ipotesi nulla, se $H=1 \Rightarrow$ rigetta l'ipotesi nulla);

- **P:** p-value, è un valore compreso nell'intervallo (0;1) che indica se la distribuzione empirica segue quella nota;
- **K:** valore del test.

Come in precedenza, i valori ottenuti per ogni distribuzione nota sono i seguenti:

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

H =
logical

1

P =
6.7992e-05

K =
0.1488
```

Figura 5.15: K-S Test distribuzione esponenziale - Mercury

```
>> [H,P,K] = kstest2(empRel, exponential2(t))

H =
logical

0

P =
0.7105

K =
0.0460
```

Figura 5.16: K-S Test distribuzione iperesponenziale - Mercury

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

H =
logical

0

P =
0.0678

K =
0.0853
```

Figura 5.17: K-S Test distribuzione Weibull - Mercury

I risultati ottenuti sono riportati nella seguente tabella.

Distribuzione	H	P	K
Esponenziale	1	0.00007	0.1488
Iperesponenziale	0	0.7105	0.0460
Weibull	0	0.0678	0.0853

Tabella 5.1: Tabella parametri K-S Test

Dalla tabella si nota che il test rigetta l'ipotesi per quanto riguarda l'esponenziale ma l'accetta per quanto riguarda l'iperesponenziale e la Weibull. Tra questi due possiamo scegliere l'iperesponenziale poiché mostra un p-value superiore a quello della Weibull.

Poiché la distribuzione di riferimento è quella iperesponenziale, si può intuire che la reliability del sistema Mercury dipende da fallimenti derivanti dall'hardware.

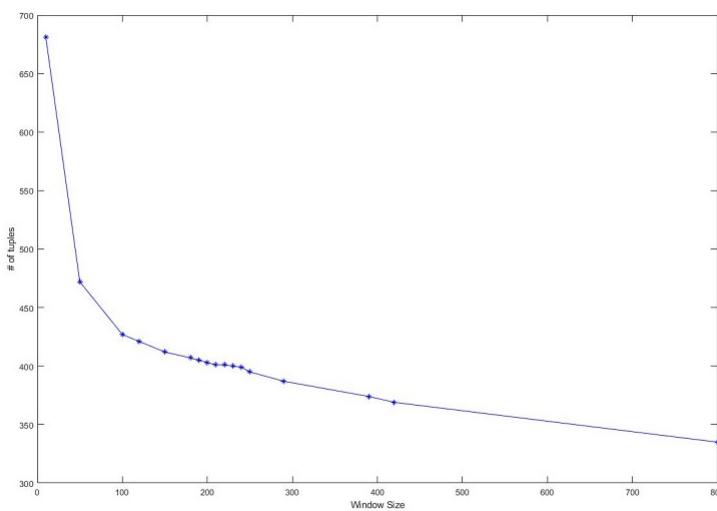


5.7 Data Manipulation - Blue Gene

Si rimanda al capitolo precedente per considerazioni sull'analisi della finestra di coalescenza.

5.7.1 Sensitivity Analysis - Blue Gene

Come fatto in precedenza nel caso di Mercury, il primo passo consiste nella ricerca del valore della finestra di coalescenza. Si è eseguito nuovamente lo script *SensitivityTuple.m* con i valori di finestra di coalescenza presenti nel file *tentative-Cwin.txt*. Il risultato è visualizzato nella figura seguente.



generando

Figura 5.18: Analisi di sensitività - BGL

Come si è visto per Mercury, si seleziona, tra tutti i valori di W calcolati, quello che produce il decremento più marcato, ossia il cosiddetto valore di ginocchio. Quindi, nel caso di esame si è scelto un valore di 180 secondi che presenta un numero di tuple pari a 407.

Successivamente tale valore è stato utilizzato per lo script *tupling_with_CWIN.sh* che analizza il file di log per produrre in output le tuple e file utili per successive analisi. Nel dettaglio:

- **lengths.txt**: indica la durata dei fallimenti e di conseguenza la lunghezza delle singole tuple.
- **interrivals.txt**: sono i tempi di interarrivo tra le tuple (TTF).
- **startingPoints.txt**: timestamp del punto di inizio della tupla.

Grazie al file *interrivals.txt* sarà possibile realizzare un'analisi sulla reliability del sistema.

Prima di andare alla fase successiva si fa anche in questo caso un'analisi di troncamenti e collisioni data la finestra di coalescenza scelta. Grazie all'aiuto di uno script Python si vanno ad analizzare le tuple che presentano i problemi descritti in precedenza.

Possiamo vedere un esempio di troncamento nella figura successiva:

The image shows two separate terminal windows side-by-side. Both windows have a title bar with the file name, a close button, and a plus sign for opening new tabs. The top window is titled "tuple_17.txt" and contains five lines of log entries. The bottom window is titled "tuple_18.txt" and also contains five lines of log entries. The log entries are truncated at the end, showing only the first few characters of each line.

```

tuple_17.txt
File Modifica Visualizza
1128951960 R44-M1-N1 J09-U11 Target=ido://FF:F2:9F:16:BB:5E:00:0D:60:E9:44:A1/JTAG/0 Message=ScomError
1128952075 R44-M1-N1 J09-U11 Target=ido://FF:F2:9F:16:BB:5E:00:0D:60:E9:44:A1/JTAG/0 Message=Invalid JtagId =
ffffffff
1128952191 R44-M1-N1 J09-U11 Target=ido://FF:F2:9F:16:BB:5E:00:0D:60:E9:44:A1/JTAG/0 Message=Invalid JtagId =
ffffffff
1128952306 R44-M1-N1 J09-U11 Target=ido://FF:F2:9F:16:BB:5E:00:0D:60:E9:44:A1/JTAG/0 Message=Invalid JtagId =
ffffffff

tuple_18.txt
File Modifica Visualizza
1128954084 R44-M1-N1 J09-U11 Target=ido://FF:F2:9F:16:BB:5E:00:0D:60:E9:44:A1/JTAG/0 Message=Invalid JtagId =
ffffffff
1128954170 R44-M1-N1 J09-U11 Target=ido://FF:F2:9F:16:BB:5E:00:0D:60:E9:44:A1/JTAG/0 Message=Invalid JtagId =
ffffffff
1128954255 R44-M1-N1 J09-U11 Target=ido://FF:F2:9F:16:BB:5E:00:0D:60:E9:44:A1/JTAG/0 Message=Invalid JtagId =
ffffffff
1128954340 R44-M1-N1 J09-U11 Target=ido://FF:F2:9F:16:BB:5E:00:0D:60:E9:44:A1/JTAG/0 Message=Invalid JtagId =
ffffffff

```

Figura 5.19: Troncamento tuple 17-18

Come in Mercury, il guasto appartiene allo stesso nodo e alla stessa categoria e di conseguenza il raggruppamento è fallito a causa della grandezza della finestra.

Per quanto riguarda le collisioni si rilevano diverse tuple con nodi diversi, tuttavia come si evince nelle figure successive le schede sono tutte di tipo I/O (J18):

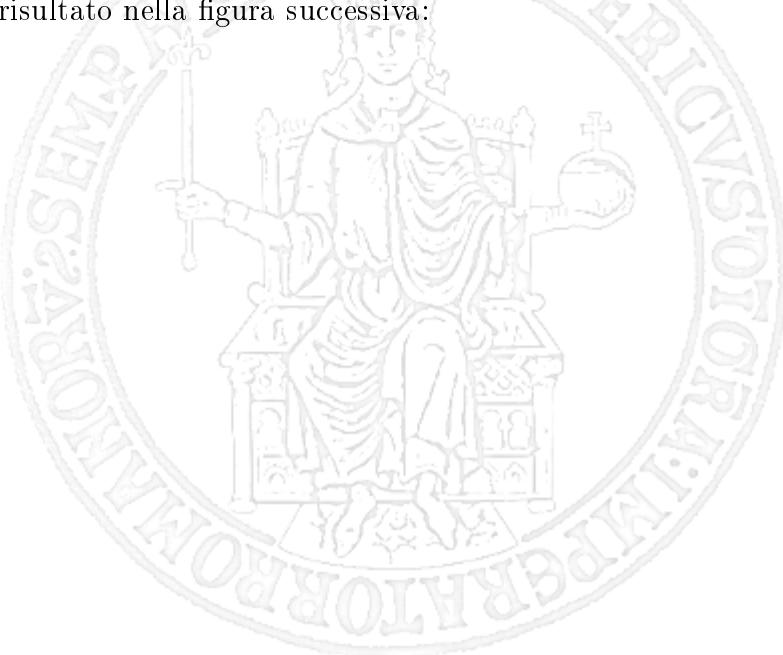
The image shows a single terminal window with a title bar "tuple_1.txt". The window displays a list of log entries. A specific entry is highlighted with a red box, showing a collision where multiple tuples (J18-U01) are associated with the same event (Lustre mount FAILED). The log entries are as follows:

```

tuple_1.txt
File Modifica Visualizza
1128621350 R00-M0-NC J18-U01 Lustre mount FAILED : bglio2 : block_id : location
1128621350 R01-M1-NC J18-U11 Lustre mount FAILED : bglio21 : block_id : location
1128621351 R07-M0-NC J18-U01 Lustre mount FAILED : bglio124 : block_id : location
1128621351 R00-M0-N4 J18-U01 Lustre mount FAILED : bglio4 : block_id : location
1128621351 R02-M0-NC J18-U01 Lustre mount FAILED : bglio36 : block_id : location
1128621351 R01-M0-N8 J18-U11 Lustre mount FAILED : bglio25 : block_id : location
1128621351 R06-M0-NC J18-U01 Lustre mount FAILED : bglio98 : block_id : location
1128621351 R04-M0-NC J18-U01 Lustre mount FAILED : bglio66 : block_id : location
1128621352 R05-M0-NC J18-U11 Lustre mount FAILED : bglio91 : block_id : location
1128621352 R03-M0-NC J18-U11 Lustre mount FAILED : bglio59 : block_id : location
1128621352 R03-M0-NC J18-U01 Lustre mount FAILED : bglio60 : block_id : location
1128621352 R03-M0-N8 J18-U11 Lustre mount FAILED : bglio57 : block_id : location
1128621352 R05-M0-NC J18-U11 Lustre mount FAILED : bglio89 : block_id : location
1128621352 R05-M1-NC J18-U01 Lustre mount FAILED : bglio88 : block_id : location
1128621352 R06-M0-NC J18-U01 Lustre mount FAILED : bglio108 : block_id : location
1128621353 R00-M0-NC J18-U11 Lustre mount FAILED : bglio11 : block_id : location
1128621353 R00-M1-NC J18-U01 Lustre mount FAILED : bglio6 : block_id : location
1128621353 R00-M1-N4 J18-U11 Lustre mount FAILED : bglio7 : block_id : location
1128621353 R03-M1-NC J18-U01 Lustre mount FAILED : bglio64 : block_id : location
1128621353 R02-M1-NC J18-U11 Lustre mount FAILED : bglio47 : block_id : location
1128621353 R04-M1-N8 J18-U01 Lustre mount FAILED : bglio78 : block_id : location
1128621353 R04-M0-NC J18-U11 Lustre mount FAILED : bglio73 : block_id : location
1128621353 R04-M0-N8 J18-U01 Lustre mount FAILED : bglio74 : block_id : location
1128621354 R05-M1-NC J18-U11 Lustre mount FAILED : bglio95 : block_id : location
1128621354 R02-M1-NC J18-U11 Lustre mount FAILED : bglio37 : block_id : location
1128621354 R06-M0-NC J18-U11 Lustre mount FAILED : bglio99 : block_id : location
1128621354 R04-M0-NC J18-U11 Lustre mount FAILED : bglio67 : block_id : location
1128621354 R00-M0-N8 J18-U01 Lustre mount FAILED : bglio10 : block_id : location
1128621354 R01-M1-NC J18-U01 Lustre mount FAILED : bglio24 : block_id : location

```

Figura 5.20: Collisione tupla 1



```

tuple_102.txt

File Modifica Visualizza

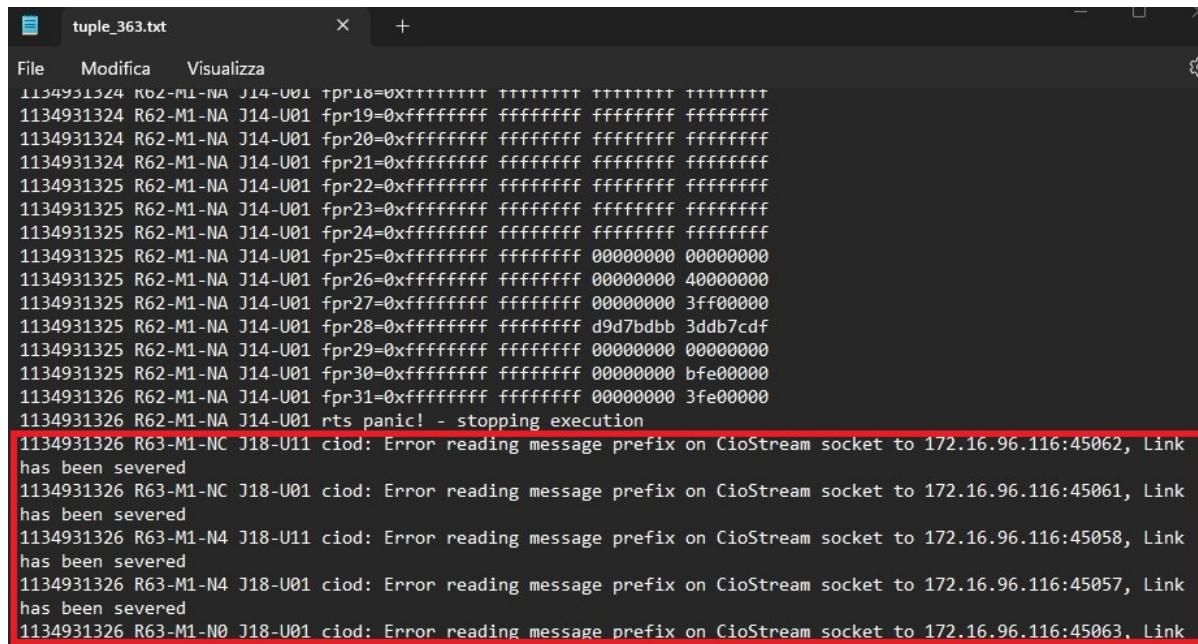
1130570696 R65-M0-N4 J18-U11 ciod: LOGIN chdir(RUNS/3Bvn_fmm/rs0000) failed: No such file or directory
1130570696 R65-M0-N4 J18-U01 ciod: LOGIN chdir(RUNS/3Bvn_fmm/rs0000) failed: No such file or directory
1130570696 R65-M1-N4 J18-U11 ciod: LOGIN chdir(RUNS/3Bvn_fmm/rs0000) failed: No such file or directory
1130570696 R65-M1-N4 J18-U01 ciod: LOGIN chdir(RUNS/3Bvn_fmm/rs0000) failed: No such file or directory
1130570696 R64-M1-NC J18-U11 ciod: LOGIN chdir(RUNS/3Bvn_fmm/rs0000) failed: No such file or directory
1130570696 R64-M1-NC J18-U01 ciod: LOGIN chdir(RUNS/3Bvn_fmm/rs0000) failed: No such file or directory
1130570696 R64-M1-N8 J18-U11 ciod: LOGIN chdir(RUNS/3Bvn_fmm/rs0000) failed: No such file or directory
1130570697 R64-M1-N8 J18-U01 ciod: LOGIN chdir(RUNS/3Bvn_fmm/rs0000) failed: No such file or directory
1130570697 R64-M1-N4 J18-U11 ciod: LOGIN chdir(RUNS/3Bvn_fmm/rs0000) failed: No such file or directory
1130570697 R64-M1-N4 J18-U01 ciod: LOGIN chdir(RUNS/3Bvn_fmm/rs0000) failed: No such file or directory
1130570697 R60-M1-NC J18-U11 ciod: LOGIN chdir(RUNS/3Bvn_fmm/rs0000) failed: No such file or directory
1130570697 R60-M1-NC J18-U01 ciod: LOGIN chdir(RUNS/3Bvn_fmm/rs0000) failed: No such file or directory
1130570697 R64-M0-NC J18-U11 ciod: LOGIN chdir(RUNS/3Bvn_fmm/rs0000) failed: No such file or directory
1130570697 R64-M0-NC J18-U01 ciod: LOGIN chdir(RUNS/3Bvn_fmm/rs0000) failed: No such file or directory
1130570697 R64-M1-N0 J18-U11 ciod: LOGIN chdir(RUNS/3Bvn_fmm/rs0000) failed: No such file or directory
1130570697 R64-M1-N0 J18-U01 ciod: LOGIN chdir(RUNS/3Bvn_fmm/rs0000) failed: No such file or directory
1130570697 R64-M0-N8 J18-U11 ciod: LOGIN chdir(RUNS/3Bvn_fmm/rs0000) failed: No such file or directory
1130570697 R64-M0-N8 J18-U01 ciod: LOGIN chdir(RUNS/3Bvn_fmm/rs0000) failed: No such file or directory
1130570697 R65-M1-N0 J18-U11 ciod: LOGIN chdir(RUNS/3Bvn_fmm/rs0000) failed: No such file or directory
1130570697 R65-M1-N0 J18-U01 ciod: LOGIN chdir(RUNS/3Bvn_fmm/rs0000) failed: No such file or directory
1130570697 R65-M1-N8 J18-U11 ciod: LOGIN chdir(RUNS/3Bvn_fmm/rs0000) failed: No such file or directory
1130570697 R65-M1-N8 J18-U01 ciod: LOGIN chdir(RUNS/3Bvn_fmm/rs0000) failed: No such file or directory
1130570698 R65-M0-N8 J18-U11 ciod: LOGIN chdir(RUNS/3Bvn_fmm/rs0000) failed: No such file or directory
1130570698 R64-M0-N0 J18-U11 ciod: LOGIN chdir(RUNS/3Bvn_fmm/rs0000) failed: No such file or directory
1130570698 R64-M0-N0 J18-U01 ciod: LOGIN chdir(RUNS/3Bvn_fmm/rs0000) failed: No such file or directory

```

Figura 5.21: Collisione tupla 102

Si può ipotizzare che un guasto di un certo tipo si sia propagato in nodi diversi causando errori rilevati e memorizzati nel Log in un determinato intertempo, quindi non è necessariamente una collisione causata da una finestra di coalescenza troppo grande.

Per poter rilevare collisioni in modo più preciso, lo script è stato modificato andando a ricercare tuple con diversi nodi e diversi tipi di scheda ossia di tipo I/O (J18) e computazionale (JX). Possiamo vedere un risultato nella figura successiva:



```

tuple_363.txt

File Modifica Visualizza
1134931324 R62-M1-NA J14-U01 fpr18=0xffffffff ffffffff ffffffff ffffffff
1134931324 R62-M1-NA J14-U01 fpr19=0xffffffff ffffffff ffffffff ffffffff
1134931324 R62-M1-NA J14-U01 fpr20=0xffffffff ffffffff ffffffff ffffffff
1134931324 R62-M1-NA J14-U01 fpr21=0xffffffff ffffffff ffffffff ffffffff
1134931325 R62-M1-NA J14-U01 fpr22=0xffffffff ffffffff ffffffff ffffffff
1134931325 R62-M1-NA J14-U01 fpr23=0xffffffff ffffffff ffffffff ffffffff
1134931325 R62-M1-NA J14-U01 fpr24=0xffffffff ffffffff ffffffff ffffffff
1134931325 R62-M1-NA J14-U01 fpr25=0xffffffff ffffffff 00000000 00000000
1134931325 R62-M1-NA J14-U01 fpr26=0xffffffff ffffffff 00000000 40000000
1134931325 R62-M1-NA J14-U01 fpr27=0xffffffff ffffffff 00000000 3ff00000
1134931325 R62-M1-NA J14-U01 fpr28=0xffffffff ffffffff d9d7bdbb 3ddb7cdf
1134931325 R62-M1-NA J14-U01 fpr29=0xffffffff ffffffff 00000000 00000000
1134931325 R62-M1-NA J14-U01 fpr30=0xffffffff ffffffff 00000000 bfe00000
1134931326 R62-M1-NA J14-U01 fpr31=0xffffffff ffffffff 00000000 3fe00000
1134931326 R62-M1-NA J14-U01 rts panic! - stopping execution
1134931326 R63-M1-NC J18-U11 ciod: Error reading message prefix on CioStream socket to 172.16.96.116:45062, Link
has been severed
1134931326 R63-M1-NC J18-U01 ciod: Error reading message prefix on CioStream socket to 172.16.96.116:45061, Link
has been severed
1134931326 R63-M1-N4 J18-U11 ciod: Error reading message prefix on CioStream socket to 172.16.96.116:45058, Link
has been severed
1134931326 R63-M1-N4 J18-U01 ciod: Error reading message prefix on CioStream socket to 172.16.96.116:45057, Link
has been severed
1134931326 R63-M1-N0 J18-U01 ciod: Error reading message prefix on CioStream socket to 172.16.96.116:45063, Link

```

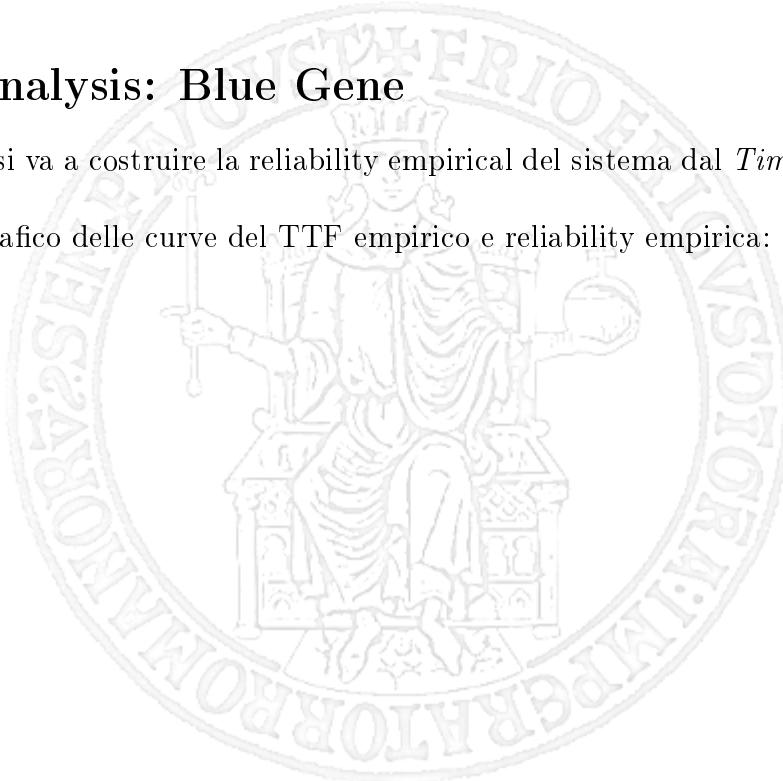
Figura 5.22: Collisione tupla 363

Nonostante siano presenti schede di diverso tipo, si può ipotizzare che il guasto di tipo computazionale generato dal primo nodo si sia propagato in nodi adiacenti mostrando errori di tipo I/O. L'ipotesi può essere confermata osservando diverse altre tuple che presentano le stesse caratteristiche.

5.8 Data Analysis: Blue Gene

Come per Mercury, si va a costruire la reliability empirical del sistema dal *Time To Failure (TTF)* come $1 - TTF$.

Viene mostrato il grafico delle curve del TTF empirico e reliability empirica:



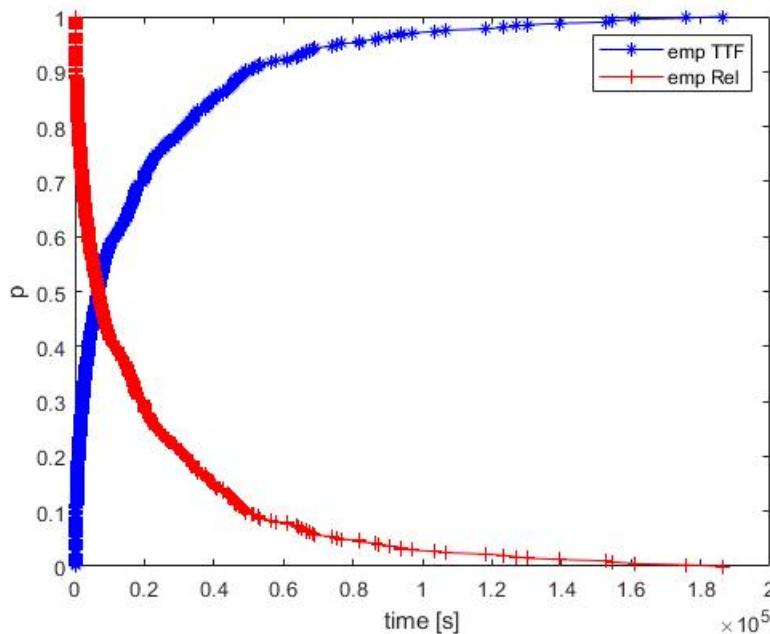


Figura 5.23: TTF e Reliability Empiriche - BGL

5.8.1 Studio della Reliability

Bisogna cercare una distribuzione teorica nota per rappresentare il più possibile la Reliability empirica trovata precedentemente. A seconda della distribuzione possiamo dedurre la tipologia dei fallimenti presenti nei dati. Utilizzando il comando Matlab *cftool* per il fitting, sono state applicate le distribuzioni viste anche nel caso di Mercury: **esponenziale, iperesponenziale, Weibull**.

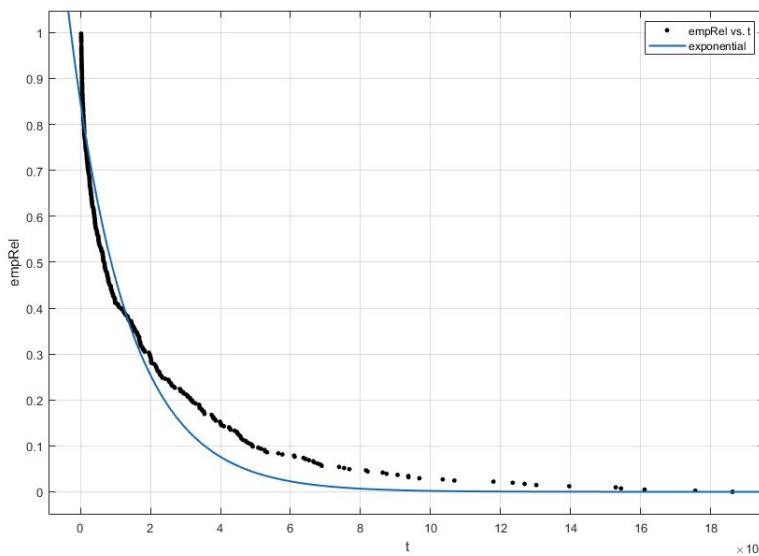


Figura 5.24: Fitting esponenziale - BGL

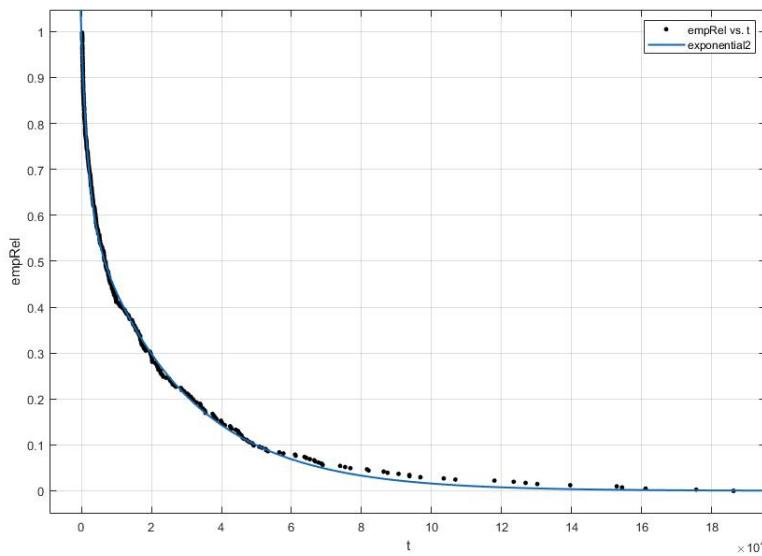


Figura 5.25: Fitting iperesponenziale - BGL

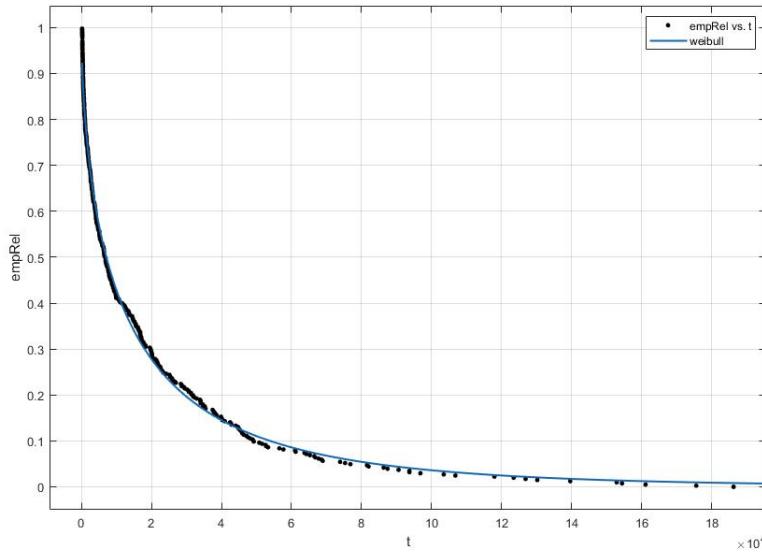


Figura 5.26: Fitting Weibull - BGL

5.8.2 Goodness of Fitting

Ricordiamo che per scegliere la migliore distribuzione notevole è possibile confrontare i valori di due parametri fondamentali:

- **SSE (Sum of Square Errors)**: somma quadratica degli errori; per avere una buona approssimazione deve essere vicina allo zero.

- **R-SQUARE:** misura il legame tra la variabilità dei dati e la correttezza del modello statistico utilizzato; deve essere il più possibile vicina a 1.

I valori ottenuti tramite *cftool* di Matlab sono i seguenti:

Table of Fits								
Fit name ▾	Data	Fit type	SSE	R-square	DFE	Adj R-sq	RMSE	# Coeff
exponential	empRel vs. t	exp1	1.4724	0.9547	396	0.9545	0.0610	2
exponential2	empRel vs. t	exp2	0.0769	0.9976	394	0.9976	0.0140	4
weibull	empRel vs. t	exp(-(b*x)^c)	0.1428	0.9956	396	0.9956	0.0190	2

Figura 5.27: Table of Fits - BGL

5.8.2.1 Test di Kolmogorov-Smirnov

Anche in questo caso si è eseguito il test utilizzando il comando Matlab *kstest2* con in input il vettore empirico e il vettore stimato. Ricordiamo che in output si ottengono i seguenti parametri:

- **H:** indica se il test di ipotesi nulla è rigettato (se $H=0 \Rightarrow$ accetta l'ipotesi nulla, se $H=1 \Rightarrow$ rigetta l'ipotesi nulla);
- **P:** p-value, è un valore compreso nell'intervallo (0;1) che indica se la distribuzione empirica segue quella nota;
- **K:** valore del test.

I valori ottenuti sono i seguenti:

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

H =
logical

1

P =
2.7130e-04

K =
0.1482
```

Figura 5.28: K-S Test distribuzione esponenziale - BGL

```

>> [H,P,K] = kstest2(empRel, exponential2(t))

H =
logical

0

P =
0.8016

K =
0.0452

```

Figura 5.29: K-S Test distribuzione iperesponenziale - BGL

```

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

H =
logical

0

P =
0.2685

K =
0.0704

```

Figura 5.30: K-S Test distribuzione Weibull - BGL

Di seguito la tabella con i risultati ottenuti.

Distribuzione	H	P	K
Esponenziale	1	0.00027	0.1482
Iperesponenziale	0	0.8016	0.0452
Weibull	0	0.2685	0.0704

Tabella 5.2: Tabella parametri K-S Test

Come Mercury, dalla tabella si nota che il test rigetta l'ipotesi per quanto riguarda l'esponenziale ma l'accetta per quanto riguarda l'iperesponenziale e la Weibull. Tra questi due possiamo scegliere l'iperesponenziale poiché mostra un p-value superiore a quello della Weibull, così come fatto nel caso di Mercury.

Anche in questo caso si può intuire che la reliability del sistema dipende da fallimenti derivanti dall'hardware.

5.9 Confronto Mercury e BlueGene

Ora è possibile confrontare i due sistemi. Come abbiamo visto, le curve della reliability possono essere approssimate con il 95% di confidenza alla distribuzione iperesponenziale. Andando a con-

frontare le due curve di reliability si puo notare come BlueGene sia più reliable di Mercury poiché quest'ultimo fallisce molto più velocemente.

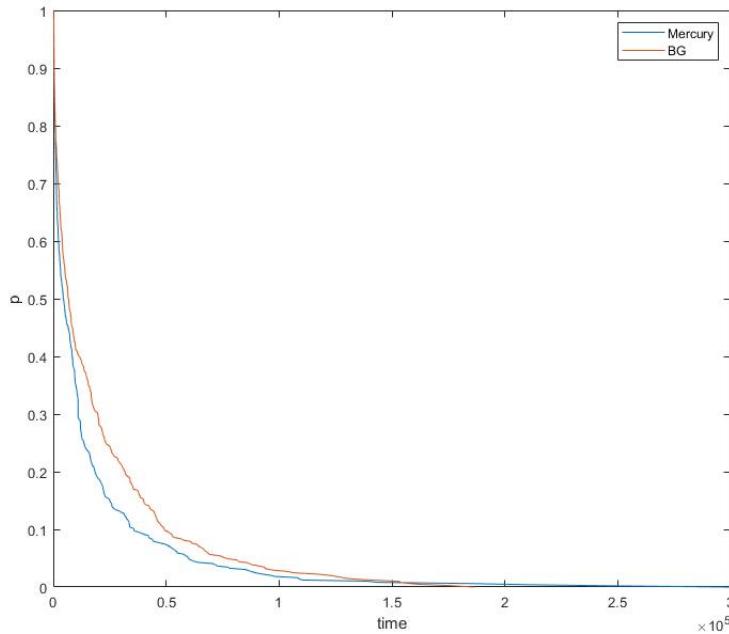


Figura 5.31: Confronto Mercury e BGL

5.10 Considerazioni Finali

5.10.1 Può la stessa finestra di coalescenza essere usata per differenti nodi (Mercury, BG/L) e differenti errori di categoria (Mercury)?

Attraverso l'analisi dello script *logStatistics.sh* è stato possibile maturare una visione d'insieme in merito ai log dei due supercalcolatori, in modo da individuare le categorie di errore ed i primi nodi più fallimentari.

```
-- 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 5.32: Analisi Statistica Mercury



```

== Total error entries ==
125624

== Breakup by CATEGORY ==
J18-U11 50055
J18-U01 49932
J14-U01 2257
J12-U01 1877
J07-U01 1780
J10-U11 1333
J03-U11 1020
J16-U11 973
J06-U11 960
J11-U11 888
J17-U11 824
J09-U01 808
J16-U01 753
J09-U11 746
J08-U01 741
J03-U01 701
J11-U01 700
J05-U11 670
J04-U01 655
J13-U01 647
J15-U01 633
J17-U01 602
J12-U11 596
J04-U11 593
J13-U11 563
J08-U11 560
J10-U01 554
J02-U01 524
J05-U01 456
J15-U11 454
J02-U11 449
J14-U11 445
J07-U11 445
J06-U01 430

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

```

Figura 5.33: Analisi Statistica BGL

In questo modo si è ottenuto, tramite estrazione, delle entry relative a:

- Un totale di sei categories per il sistema Mercury;
- I 10 nodi più fallimentari per Mercury;
- I 10 nodi più fallimentari per BlueGene.

5.10.1.1 Coalescence Window Mercury

Nodi Mercury

La ricerca della finestra di coalescenza per i 10 nodi è stata effettuata a partire dello script *logStatistics.sh* e per ognuno dei nodi è stato possibile ottenere un sottoinsieme dell'errorlog originario grazie allo script *filter.sh*.

In seguito sono state svolte le operazioni in maniera del tutto similiare a quanto effettuato in precedenza.

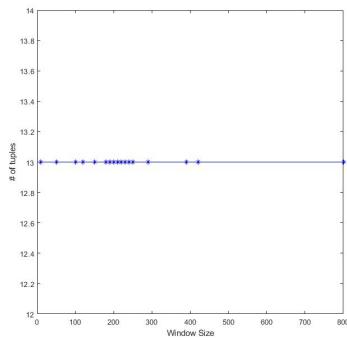


Figura 5.34: tg-c117

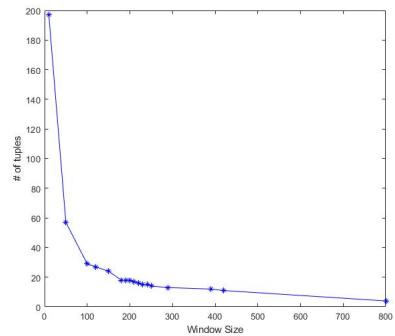


Figura 5.35: tg-c238

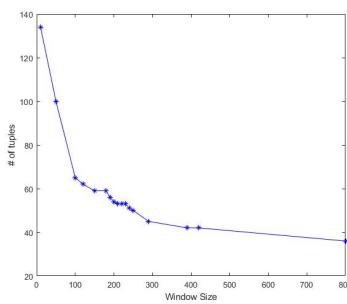


Figura 5.36: tg-c242

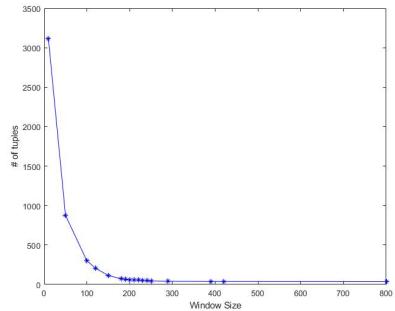


Figura 5.37: tg-c401

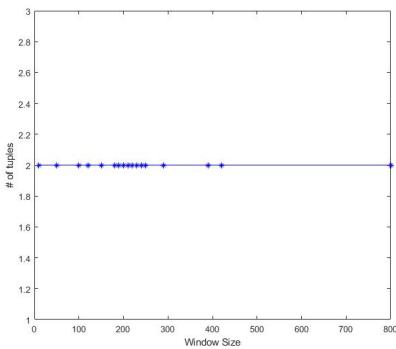


Figura 5.38: tg-c572

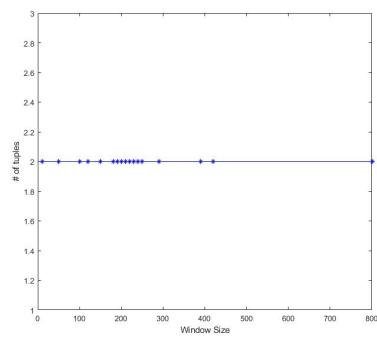


Figura 5.39: tg-c648

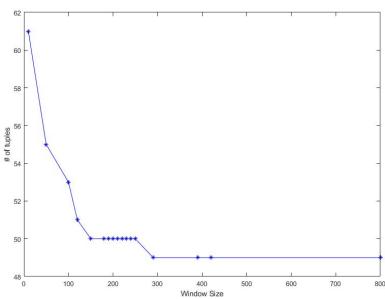


Figura 5.40: tg-c669

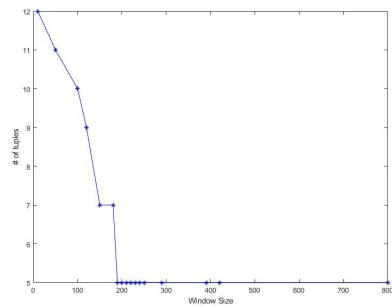


Figura 5.41: tg-login3

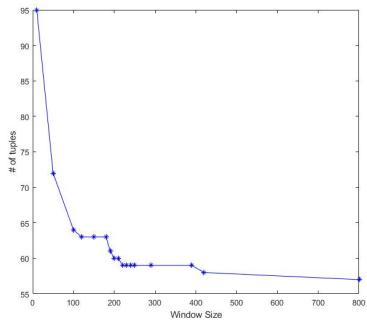


Figura 5.42: tg-master

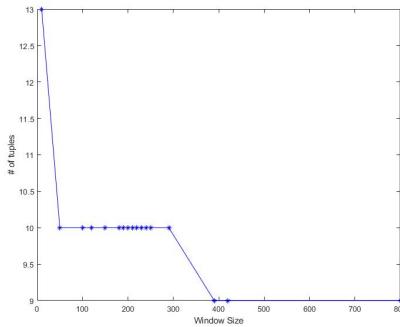


Figura 5.43: tg-s044

Nodo	Finestra	Tuple
tg-c401	180	72
tg-master	220	59
tg-c572	10	2
tg-s044	50	10
tg-c238	180	18
tg-c242	290	45
tg-c648	10	2
tg-login3	190	5
tg-c117	10	13
tg-c669	180	50

Tabella 5.3: Tabella Nodi Mercury

Concludendo l'analisi, come si può denotare dai grafici e dalla tabella, non è possibile scegliere per tutti i nodi la stessa finestra di coalescenza.

Categories Mercury

La stessa operazione per quanto riguarda il supercalcolatore Mercury è stata ripetuta per quelle che sono state le differenti categorie di errori individuate grazie allo script *logStatistics.sh*. Va considerato che un numero elevato di eventi non è sinonimo necessariamente di un elevato numero di tuple, in quanto più eventi possono essere correlati allo stesso errore. Si procede quindi all'applicazione dello script Matlab *SensitivityTuple.m*.



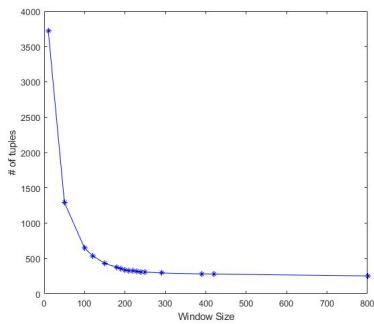


Figura 5.44: DEV

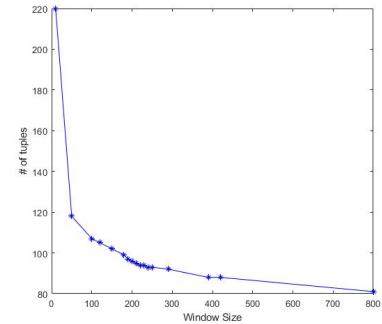


Figura 5.45: I-O

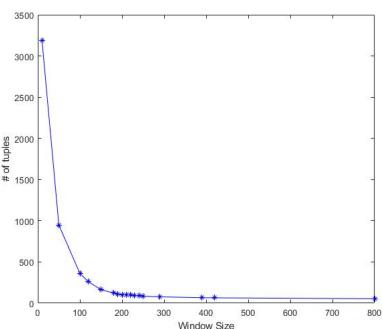


Figura 5.46: MEM

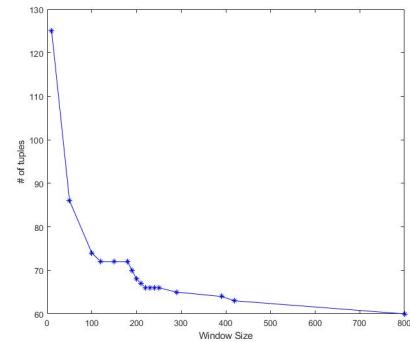


Figura 5.47: NET

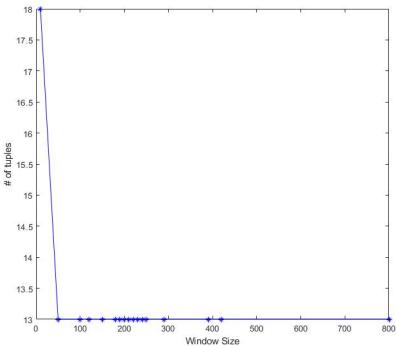


Figura 5.48: OTH

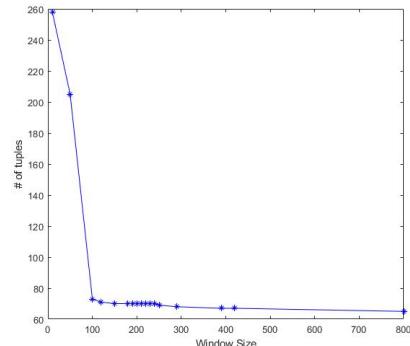


Figura 5.49: PRO

Categoria	Finestra	Tuple
DEV	220	323
MEM	200	105
I-O	240	93
NET	220	66
PRO	100	73
OTH	50	13

Tabella 5.4: Tabella Categorie Mercury

Volendo concludere, come fatto in precedenza, si traggono le stesse conclusioni andando a denotare come non è possibile scegliere la stessa finestra di coalescenza.

5.10.1.2 Coalescence Window BlueGene

Nodi BGL

L'analisi sui nodi effettuata per il sistema Mercury è stata effettuata in maniera del tutto identica per quanto riguarda i nodi più fallimentari del supercalcolatore BlueGene.

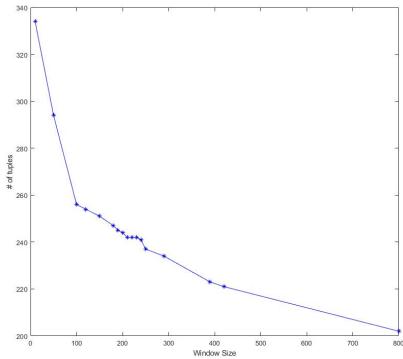


Figura 5.50: J18-U01

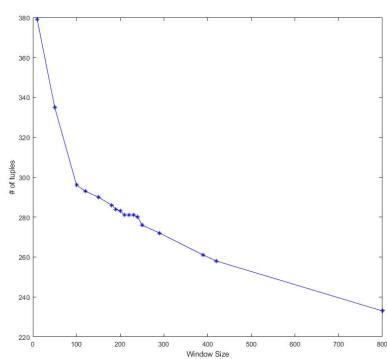


Figura 5.51: J18-U11

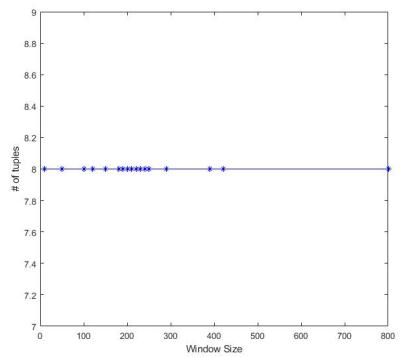


Figura 5.52: R03-M1-NF

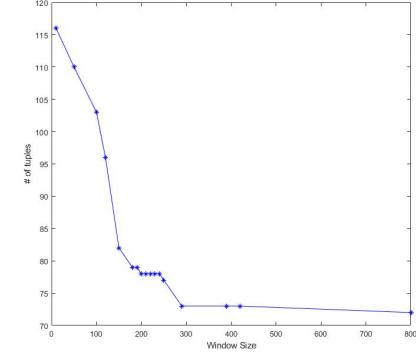


Figura 5.53: R12-M0-N0

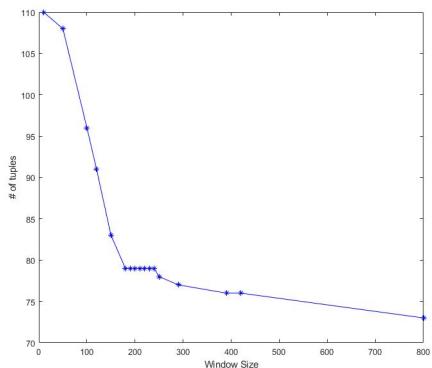


Figura 5.54: R36-M1-N0

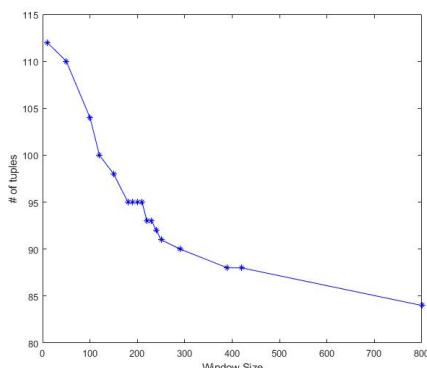


Figura 5.55: R62-M0-N4

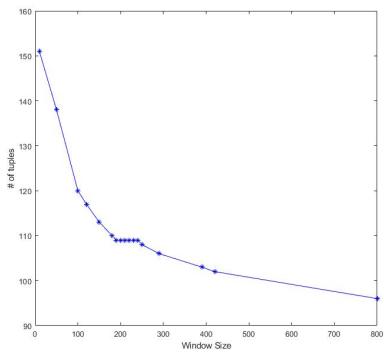


Figura 5.56: R63-M0-N0

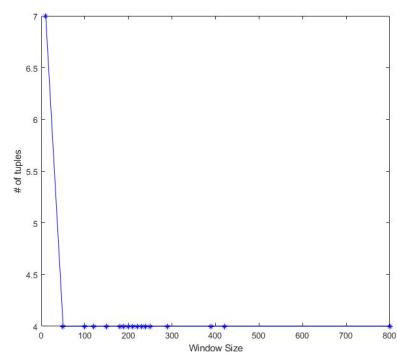


Figura 5.57: R63-M0-N2

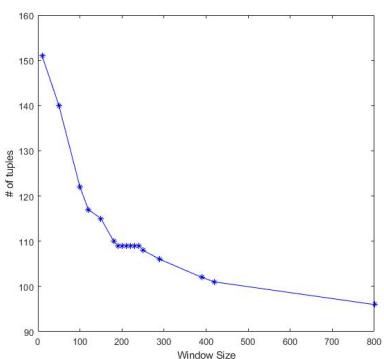


Figura 5.58: R63-M0-N4

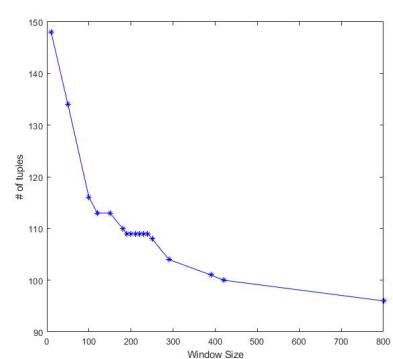


Figura 5.59: R63-M0-N8

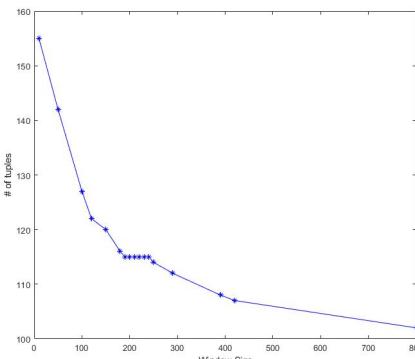


Figura 5.60: R63-M0-NC

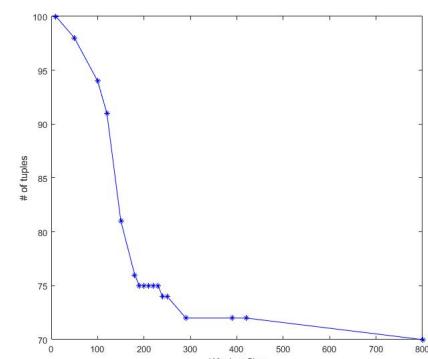


Figura 5.61: R71-M0-N4

Nodo	Finestra	Tuple
R71-M0-N4	290	72
R12-M0-N0	290	73
R63-M0-N2	50	4
R03-M1-NF	10	8
R63-M0-N0	250	108
R36-M1-N0	180	79
R62-M0-N4	250	91
R63-M0-NC	190	115
R63-M0-N8	190	109
R63-M0-N4	190	109

Tabella 5.5: Tabella Nodi BGL

Infine si deduce che non è possibile andare a scegliere la stessa finestra di coalescenza per tutti i nodi.

5.10.2 Qual è la relazione tra la reliability del sistema e dei nodi?

5.10.2.1 Mercury

Per confrontare la reliability dei singoli nodi con la reliability del sistema, nel caso di Mercury si riapplicherà lo script *filter.sh* per recuperare i log dei nodi più significativi del supercomputer e si genererà successivamente la reliability empirica per ogni nodo considerato. In questo modo sarà poi possibile rappresentare le singole curve di reliability dei nodi, così da confrontarle con quella del sistema.

Analizzando i risultati prodotti da *logStatistics.sh* si è deciso di effettuare un'analisi solo per i nodi con un numero di entries significativo. Nel nostro caso i nodi di riferimento sono 5:

- tg-c401 con finestra di coalescenza 180 e 72 tuple;
- tg-master con finestra di coalescenza 220 e 59 tuple;
- tg-c572 con finestra di coalescenza 10 e 2 tuple;
- tg-s044 con finestra di coalescenza 50 e 10 tuple;
- tg-c238 con finestra di coalescenza 180 e 18 tuple.

Si può notare, tuttavia, come il nodo tg-c572 presenti solo 2 tuple, di conseguenza avrà una curva di reliability empirica corrispondente a un singolo punto. Di seguito verranno visualizzati i grafici di reliability dei quattro nodi considerati:

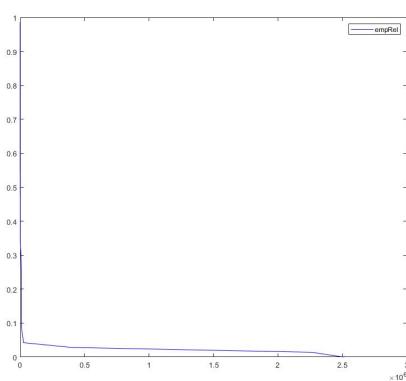


Figura 5.62: tg-c401

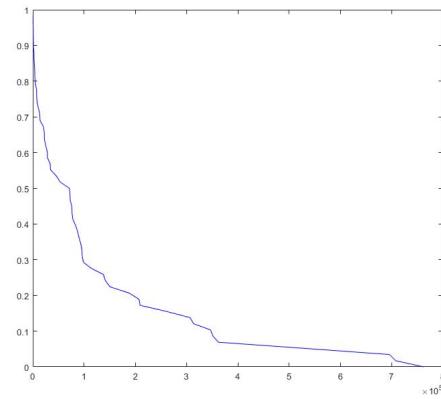


Figura 5.63: tg-master

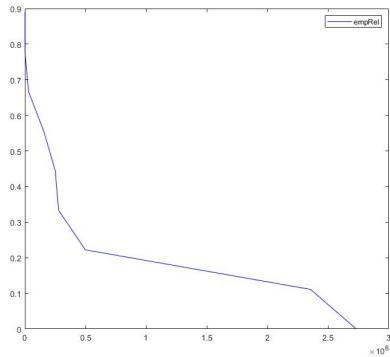


Figura 5.64: tg-s044

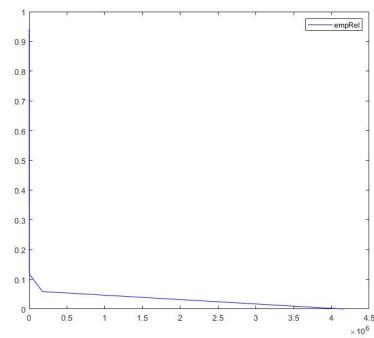
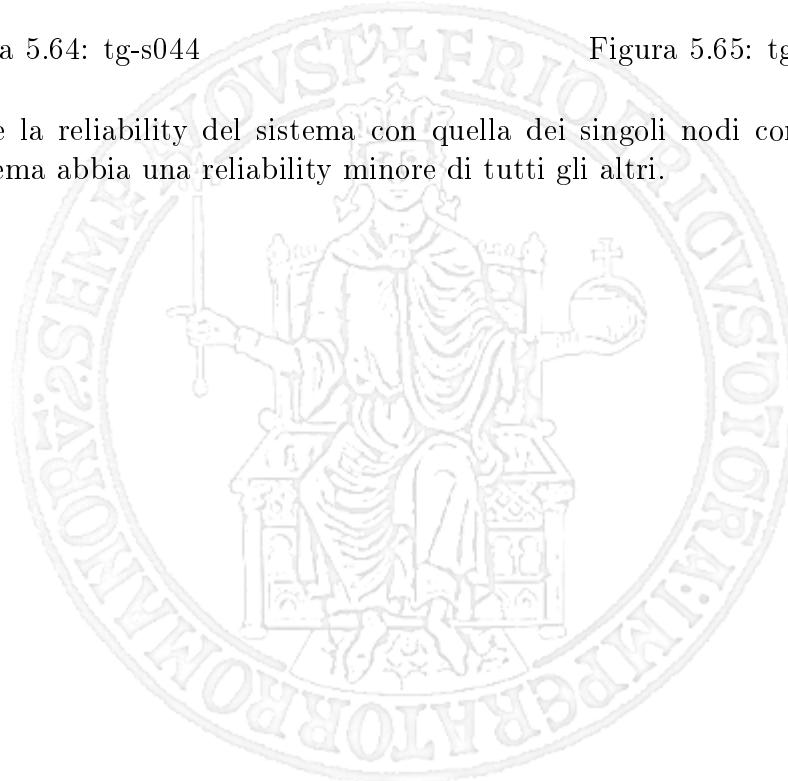


Figura 5.65: tg-c238

Confrontando infine la reliability del sistema con quella dei singoli nodi considerati è possibile osservare che il sistema abbia una reliability minore di tutti gli altri.



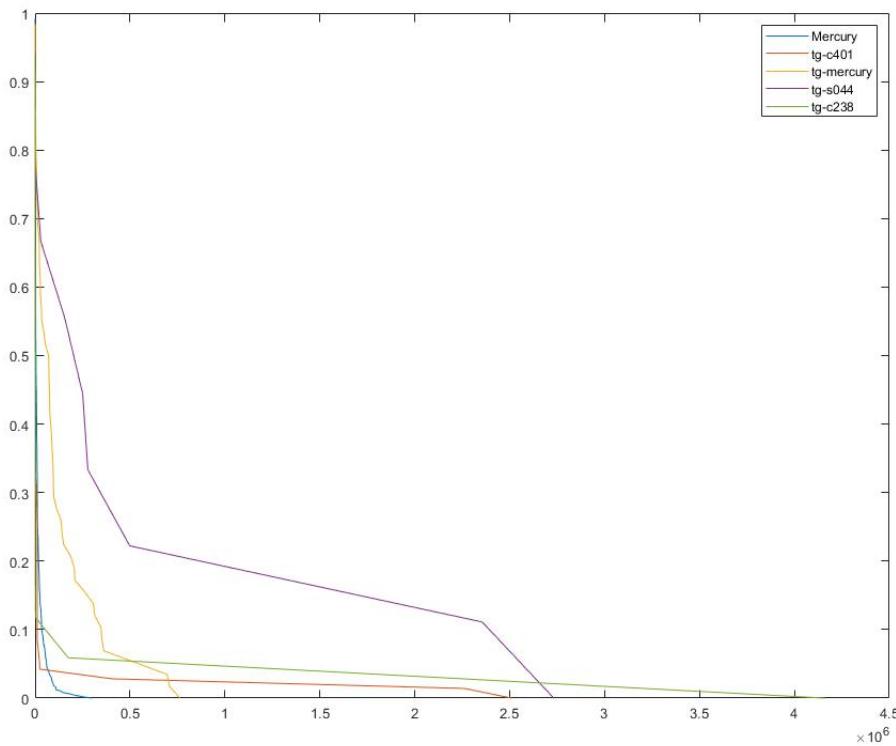


Figura 5.66: Confronto Mercury e nodi significativi

5.10.2.2 BlueGene

Per confrontare la reliability dei singoli nodi con la reliability del sistema, nel caso di BlueGene si riapplicherà lo script *filter.sh* per recuperare i log dei nodi più significativi del supercomputer e si genererà successivamente la reliability empirica per ogni nodo considerato. In questo modo sarà poi possibile rappresentare le singole curve di reliability dei nodi, così da confrontarle con quella del sistema.

Anche in questo caso, analizzando i risultati prodotti da *logStatistics.sh* si è deciso di effettuare un'analisi solo per i nodi con un numero di entries significativo. Nel nostro caso i nodi di riferimento sono 4:

- R71-M0-N4 con finestra di coalescenza 290 e 72 tuple;
- R12-M0-N0 con finestra di coalescenza 290 e 73 tuple;
- R63-M0-N2 con finestra di coalescenza 50 e 4 tuple;
- R03-M1-NF con finestra di coalescenza 10 e 8 tuple;.

Di seguito verranno visualizzati i grafici di reliability dei quattro nodi considerati:

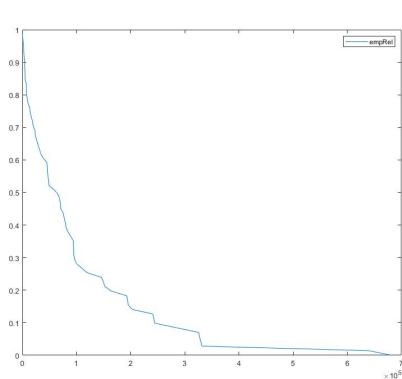


Figura 5.67: R71-M0-N4

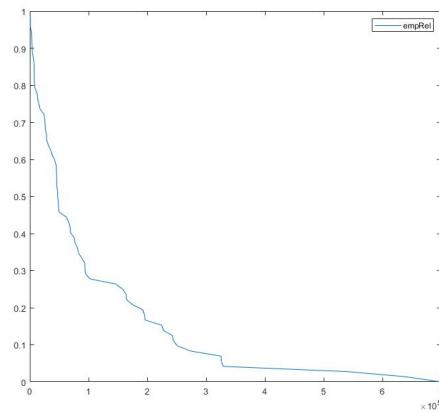


Figura 5.68: R12-M0-N0

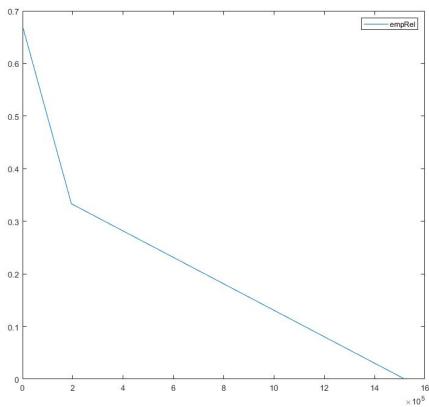


Figura 5.69: R63-M0-N2

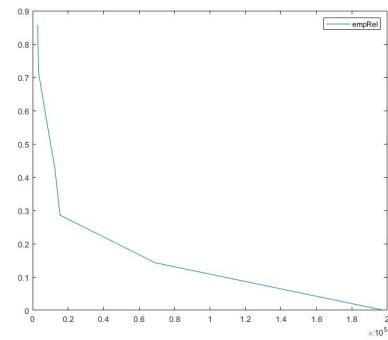


Figura 5.70: R03-M1-NF

Anche in questo caso, confrontando infine la reliability del sistema con quella dei singoli nodi considerati, è possibile osservare che il sistema abbia una reliability peggiore degli altri.

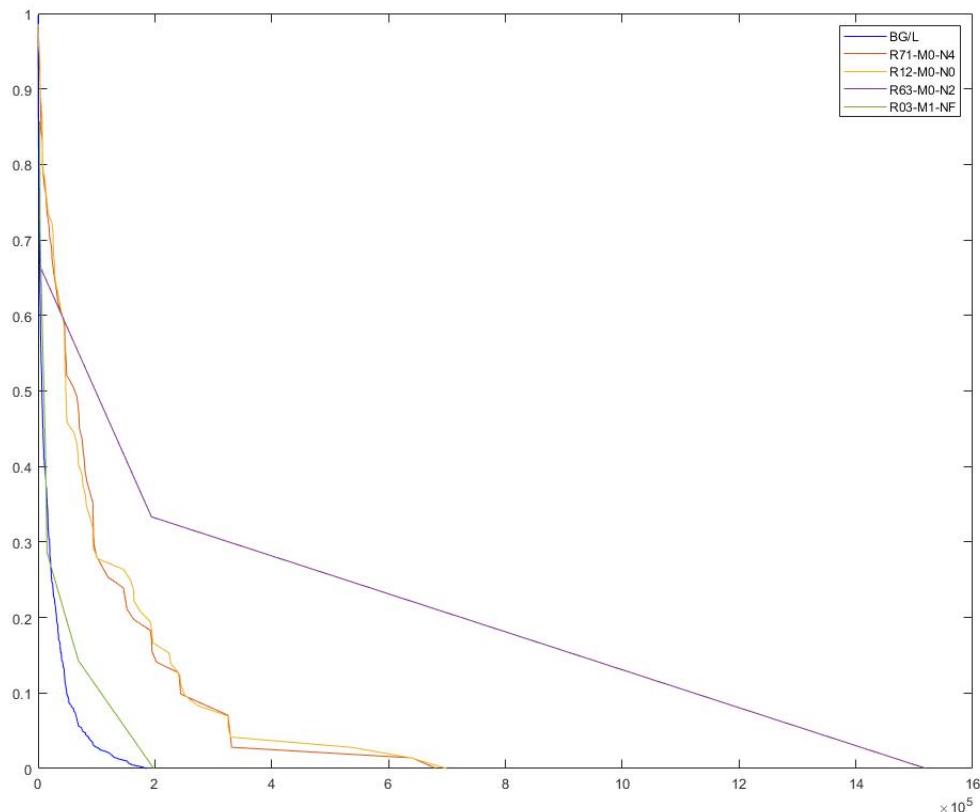


Figura 5.71: Confronto Mercury e nodi significativi

5.10.3 Possono esistere dei dependability-bottlenecks?

5.10.3.1 Mercury

Per quanto riguarda il sistema Mercury si ottiene che analizzando di nuovo l'output di *logStatistics.sh* possiamo verificare la presenza di possibili bottlenecks. Nel dettaglio, abbiamo due nodi col maggior numero di fallimenti, ossia *tg-c401* e *tg-master*. Sappiamo, date le analisi fatte in precedenza, che questi due nodi sono anche quelli con il maggior numero di tuple.

Si può notare nella figura successiva che il grafico del nodo **tg-master** ha una reliability che crolla a picco e quindi rappresenta il **bottleneck** del sistema Mercury.

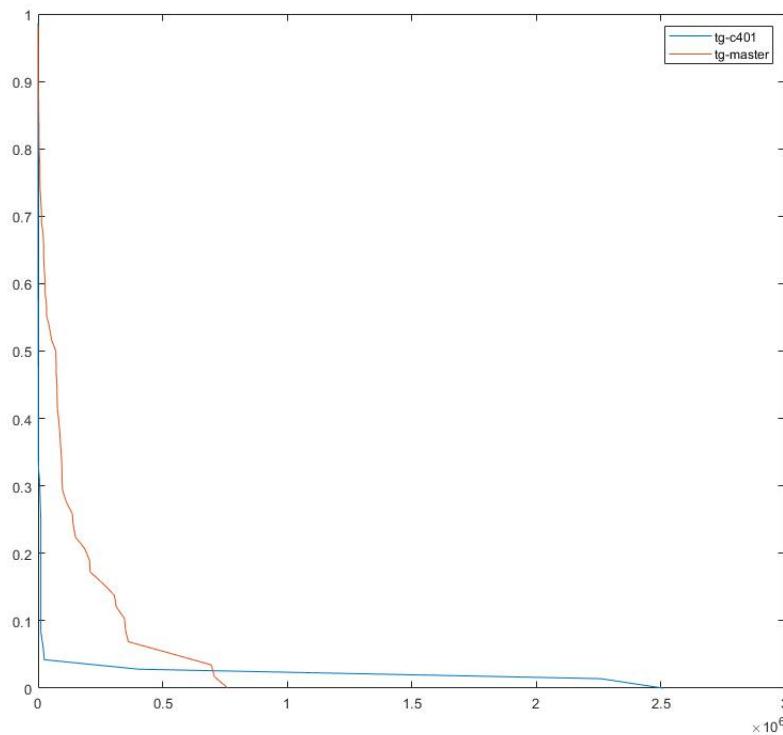


Figura 5.72: Confronto Reliability tg-c401 e tg-master - Mercury

5.10.3.2 BlueGene

Relativamente al sistema BlueGene si ottiene che analizzando di nuovo l'output di *logStatistics.sh* possiamo verificare la presenza di possibili bottlenecks. Nel dettaglio abbiamo cinque nodi che presentano il maggior numero dei fallimenti e il maggior numero di tuple tranne nei casi di *R63-M0-N2* e *R03-M1-NF*.

Si può notare nella figura successiva che il nodo **R63-M0-N0** è il **bottleneck** del sistema poiché ha una reliability leggermente più critica degli altri due considerati.

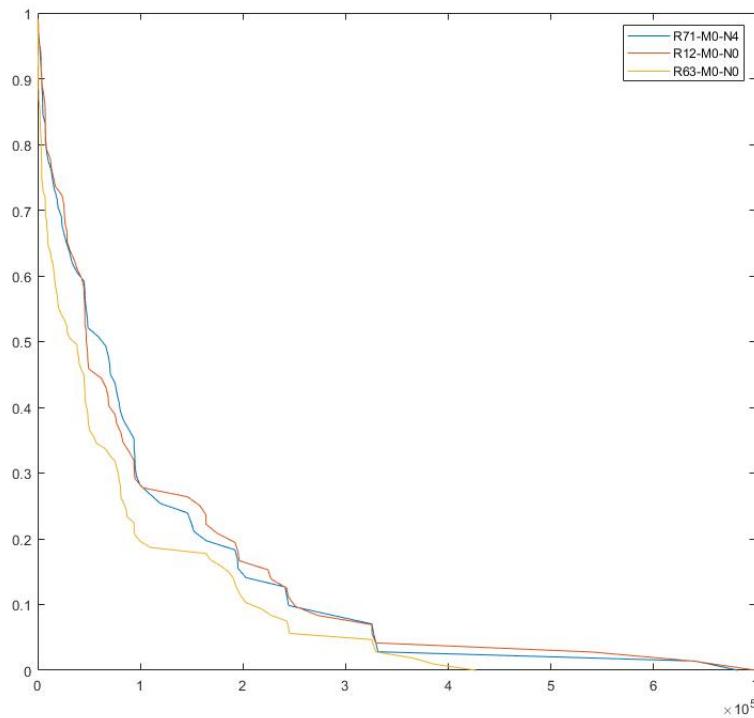


Figura 5.73: Confronto Reliability R71-M0-N4, R12-M0-N0 e R63-M0-N0 - BlueGene

5.10.4 Nodi funzionali simili hanno parametri di reliability simili?

Per rispondere a tale domanda ci si soffermerà su nodi della categoria I/O (J18) del sistema BlueGene. Come già fatto in precedenza, si va a filtrare l'analisi solo per le categorie $J18-U01$ e $J18-U11$ calcolando successivamente la reliability per ognuna. I valori della finestra di coalescenza scelti sono:

Categoria	Finestra di coalescenza	Tuple
J18-U01	250	237
J18-U11	250	276

Tabella 5.6: Finestre di coalescenza categorie BlueGene

Il fitting migliore per le categorie corrisponde all'iperesponenziale e andando a plottare le due reliability si può notare come le due categorie siano molto simili.

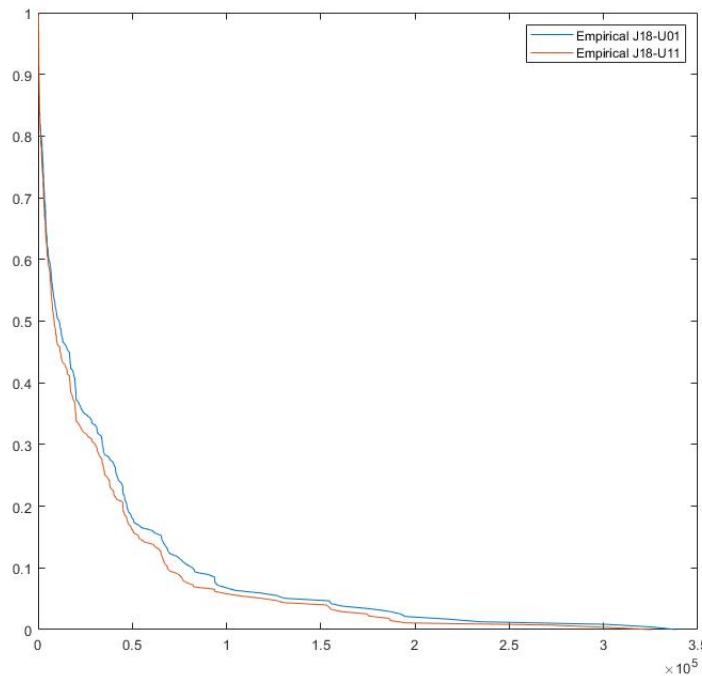
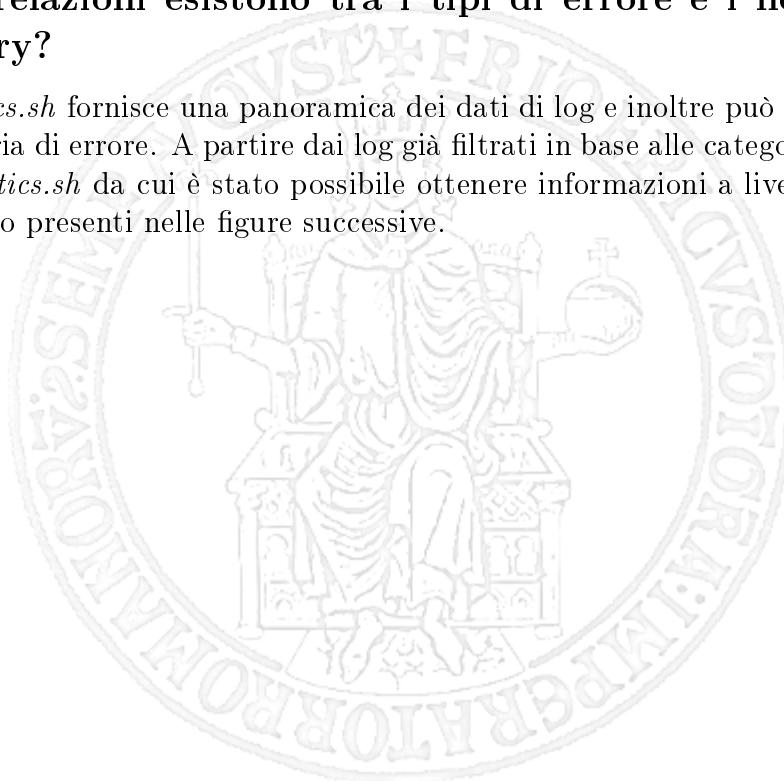


Figura 5.74: Confronto Reliability J18-U01 e J18-U11 - BlueGene

5.10.5 Quali relazioni esistono tra i tipi di errore e i nodi nel caso di Mercury?

Lo script *logStatistics.sh* fornisce una panoramica dei dati di log e inoltre può eseguire il conteggio di eventi per categoria di errore. A partire dai log già filtrati in base alle categorie, è stato di nuovo richiamato *logStatistics.sh* da cui è stato possibile ottenere informazioni a livello di categoria e di nodo. I risultati sono presenti nelle figure successive.



```

== Total error entries ==
57248

== Breakup by CATEGORY ==
DEV 57248

== Breakup by NODE* ==
tg-c401 50782
tg-c572 3176
tg-c238 1871
tg-c242 918
tg-c117 263
tg-c669 257
tg-s176 208
tg-c894 187
tg-c781 92
tg-c407 83
* only the 10 most occurring nodes are reported
    
```

Figura 5.75: CATEGORIA DEV - Mercury

```

== Total error entries ==
12819

== Breakup by CATEGORY ==
MEM 12819

== Breakup by NODE* ==
tg-c401 11558
tg-c572 845
tg-c238 197
tg-c242 149
tg-c894 28
tg-c407 14
tg-c669 10
tg-c735 6
tg-c117 5
tg-c575 3
* only the 10 most occurring nodes are reported
    
```

Figura 5.76: CATEGORIA MEM - Mercury

```

== Total error entries ==
5547

== Breakup by CATEGORY ==
I-O 5547

== Breakup by NODE* ==
tg-s044 3208
tg-master 452
tg-login3 381
tg-s038 230
tg-c550 222
tg-login4 148
tg-login1 146
tg-login2 145
tg-c685 63
tg-c860 61
* only the 10 most occurring nodes are reported
    
```

Figura 5.77: CATEGORIA I-O - Mercury

```

== Total error entries ==
3702

== Breakup by CATEGORY ==
NET 3702

== Breakup by NODE* ==
tg-master 3639
tg-c550 33
tg-c196 14
tg-c238 3
tg-s044 2
tg-c027 2
tg-s176 1
tg-s131 1
tg-login3 1
tg-login2 1
* only the 10 most occurring nodes are reported
    
```

Figura 5.78: CATEGORIA NET - Mercury

```

== Total error entries ==
1504

== Breakup by CATEGORY ==
PRO 1504

== Breakup by NODE* ==
tg-c648 616
tg-c324 239
tg-c284 180
tg-c451 159
tg-c447 136
tg-c415 61
tg-c733 28
tg-c027 22
tg-c533 18
tg-c645 9
* only the 10 most occurring nodes are reported
    
```

Figura 5.79: CATEGORIA PRO - Mercury

```

== Total error entries ==
34

== Breakup by CATEGORY ==
OTH 34

== Breakup by NODE* ==
tg-s044 14
tg-master 3
tg-c894 3
tg-c834 3
tg-c733 3
tg-c860 2
tg-c383 2
tg-c247 2
tg-c685 1
tg-c234 1
* only the 10 most occurring nodes are reported
    
```

Figura 5.80: CATEGORIA OTH - Mercury

Si riportano nelle seguenti tabella i nodi che più contribuiscono alla categoria d'errore, riportando il numero di occorrenze (in percentuale) per ogni categoria.

Nodo	DEV	MEM	I-O	NET	PRO	OTH	Totale
tg-c401	88.7%	90.1%	0%	0%	0%	0%	77.1%
tg-master	0%	0%	8.14%	92.2%	0%	8.82%	5.06%
tg-c572	5.54%	6.59%	~0%	0%	0%	0%	4.98%
tg-s044	0%	0%	57.8%	~0%	0%	41.1%	3.98%
tg-c238	1.87%	1.5%	~0%	~0%	0%	0%	1.57%

Tabella 5.7: Tabella riassuntiva Mercury

Osservando la tabella possiamo effettuare una serie di considerazioni:

- Per le categorie **DEV** e **MEM** possiamo dire che il nodo **tg-c401** presenta la maggior parte degli errori con percentuali di 88.7% e 90.1%. Inoltre gli errori sono fortemente correlati a nodi computazionali.
- Per la categoria **I-O** la percentuale maggiore di errori è presentata dal nodo di storage **tg-s044** seguito da una piccola percentuale del nodo **tg-master**.

- Quasi tutti gli errori di tipologia **NET** provengono dal nodo **tg-master** poiché è responsabile dei servizi fondamentali e di conseguenza effettua molte richieste sulla rete.
- Il tipo **PRO** presenta tutti errori provenienti dai nodi computazionali tuttavia non è rilevante ai fini dell'analisi dei principali nodi coinvolti.



Capitolo 6

Regression

6.1 Introduzione

L'obiettivo di questo homework è quello di analizzare una serie di dati temporali al fine di individuare una tendenza sottostante. Per fare ciò si studia un modello regressivo che è un tipo di modello statistico che cerca di stabilire una relazione tra una variabile di risposta (stimata) e una variabile di predizione. La relazione tra le due variabili viene stabilita attraverso una funzione lineare che descrive il modello. Se esiste una tendenza, può essere positiva, negativa o nulla e in base a questo valore si può valutare una corrispondenza tra i dati in esame.

6.2 Dataset EXP1

Il dataset è formato da:

- **Variabili di risposta:** nbytes, byte rec e byte sent;
- **Variabile di predizione:** observation.

6.2.1 Modello regressivo lineare semplice

Si applica il modello regressivo lineare semplice la cui risposta è una funzione lineare di un singolo predittore. Si va a fare una stima lineare sul software *JMP* come mostrato in figura.

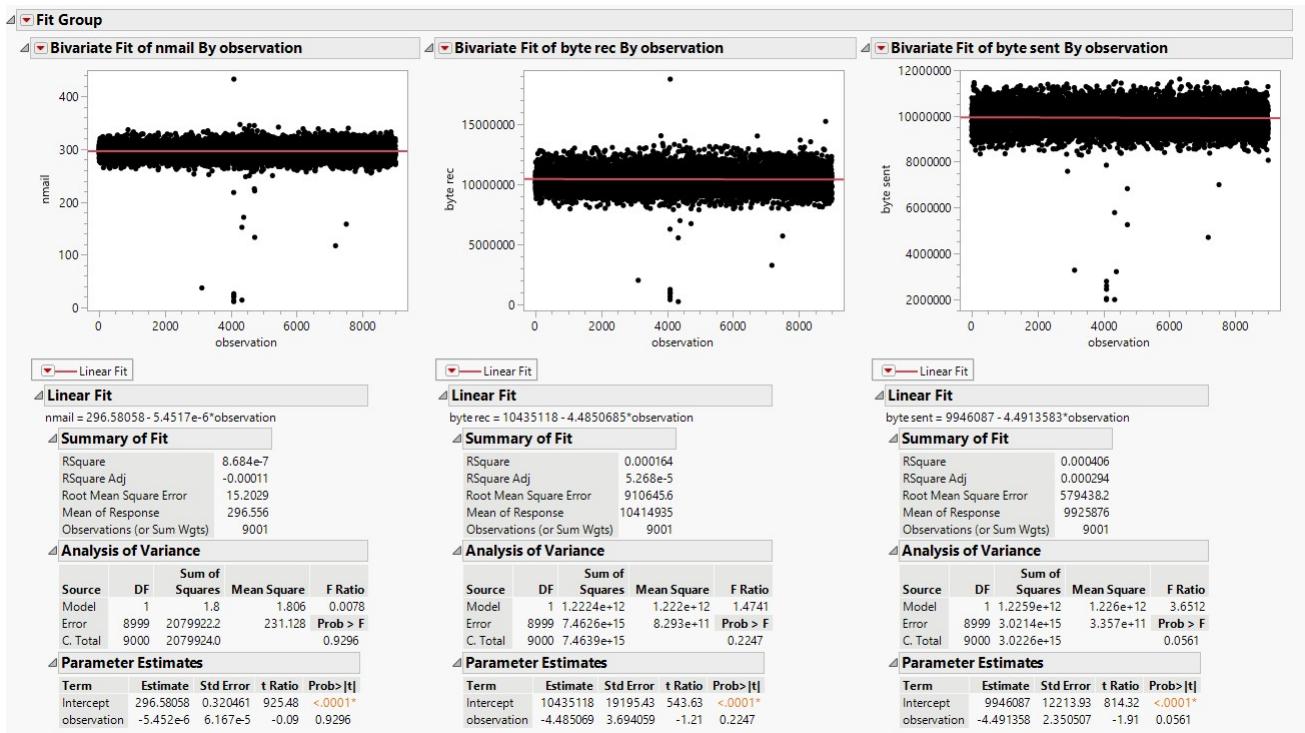


Figura 6.1: Stima lineare EXP1

Inizialmente si va a stimare il modello attraverso l'R-quadro, ovvero il coefficiente di determinazione che misura il legame tra la variabilità dei dati e la correttezza del modello statistico utilizzato, ovvero quanto si adatta ai dati analizzati. Il valore varia tra 0 e 1, dove un valore di 1 indica che il modello spiega al 100% la varianza dei dati, mentre un valore di 0 non spiega la varianza dei dati. Nel caso in esame l'R-quadro ha valori molto bassi, tuttavia lo scopo dell'homework è quello di fare trend detection e per questo motivo si possono prendere anche valori piccoli di R-quadro. Successivamente occorre verificare la significatività dei parametri della retta regressiva, in particolare della **pendenza**, tramite un opportuno test statistico.

6.2.2 Verifica delle assunzioni

Per verificare le assunzioni del modello regressivo, si può utilizzare un test visivo per validare l'ipotesi di **omoschedasticità** e **normalità** dei **residui**. Le ipotesi implicano che le differenze tra i valori previsti dal modello e i valori osservati abbiano una varianza costante per ogni valore osservato e che seguano una distribuzione normale.

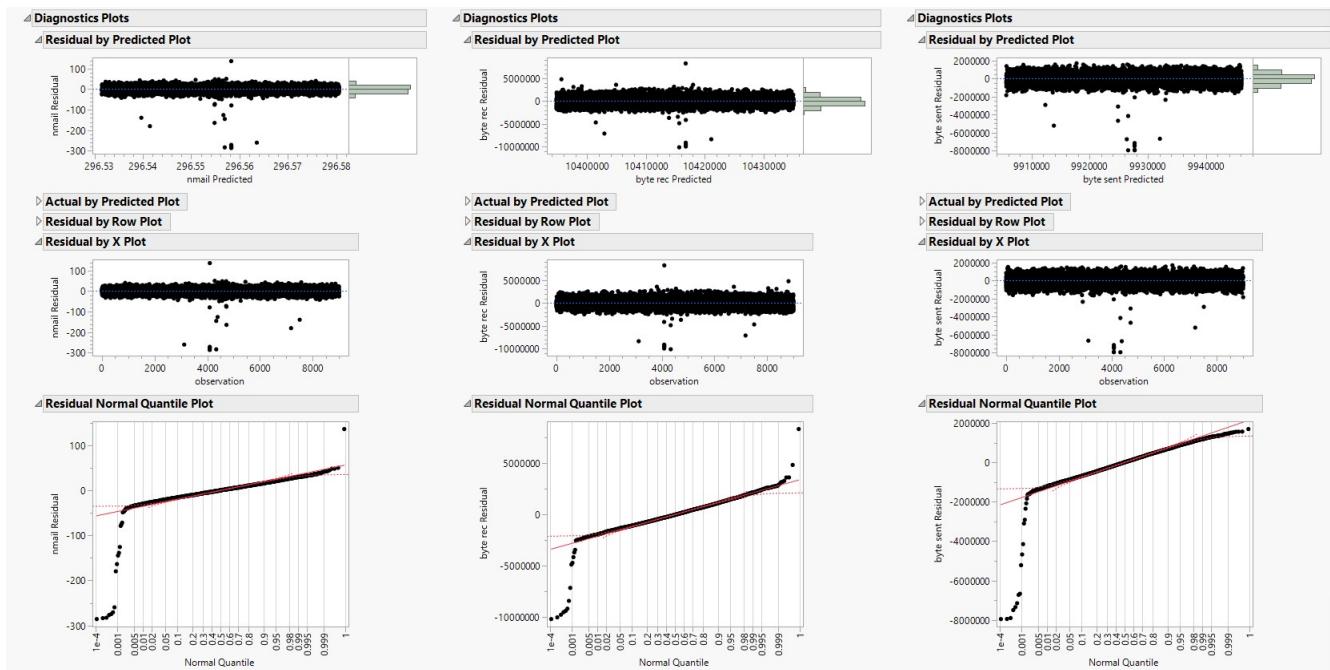


Figura 6.2: Verifica assunzioni EXP1

Come si può notare dai test visivi, l'ipotesi di omoschedasticità può essere soddisfatta poiché non sono presenti trend evidenti nei primi due grafici in figura. Tuttavia, l'ipotesi di normalità non può essere completamente soddisfatta a causa delle code presenti nel QQplot derivanti dalla presenza di outliers. Si conclude che i valori sono outliers per tutte le variabili, e quindi si decide di filtrarli. A valle del filtraggio, le assunzioni possono essere soddisfatte dunque si applica un test parametrico per la significatività della pendenza.

6.2.3 Stima dei parametri regressivi

Parameter Estimates						
Term	Estimate	Std Error	t Ratio	Prob> t	Lower 95%	Upper 95%
Intercept	296.90993	0.264421	1122.9	<.0001*	296.3916	297.4285
observation	-6.667e-6	5.088e-5	-0.13	0.8957	-0.000106	9.3069e-5

Parameter Estimates						
Term	Estimate	Std Error	t Ratio	Prob> t	Lower 95%	Upper 95%
Intercept	10446012	18188.19	574.33	<.0001*	10410359	10481665
observation	-4.499023	3.499766	-1.29	0.1986	-11.35936	2.3613169

Parameter Estimates						
Term	Estimate	Std Error	t Ratio	Prob> t	Lower 95%	Upper 95%
Intercept	99553765	11105.33	896.45	<.0001*	99336075	99771454
observation	-4.553614	2.136884	-2.13	0.0331*	-8.742393	-0.364835

Figura 6.3: Stima parametri EXP1

Dalle tabelle si può notare che nel caso di nmail e byte rec l'intervallo di confidenza al 95% relativo alla pendenza della retta di regressione include lo zero, dunque possiamo affermare che la pendenza relativa al **modello della popolazione** ha una probabilità al 95% che sia significativamente uguale a zero. Invece, nel caso di byte sent possiamo dire che è significativamente diversa da zero, dunque possiamo affermare la presenza di un trend per questa variabile.

6.3 Dataset EXP2

Il dataset è formato da:

- **Variabili di risposta:** nbytes, byte rec e byte sent;

- **Variabile di predizione:** observation.

6.3.1 Modello regressivo lineare semplice

Si applica il modello regressivo lineare semplice la cui risposta è una funzione lineare di un singolo predittore. Si va a fare una stima lineare sul software *JMP* come mostrato in figura.

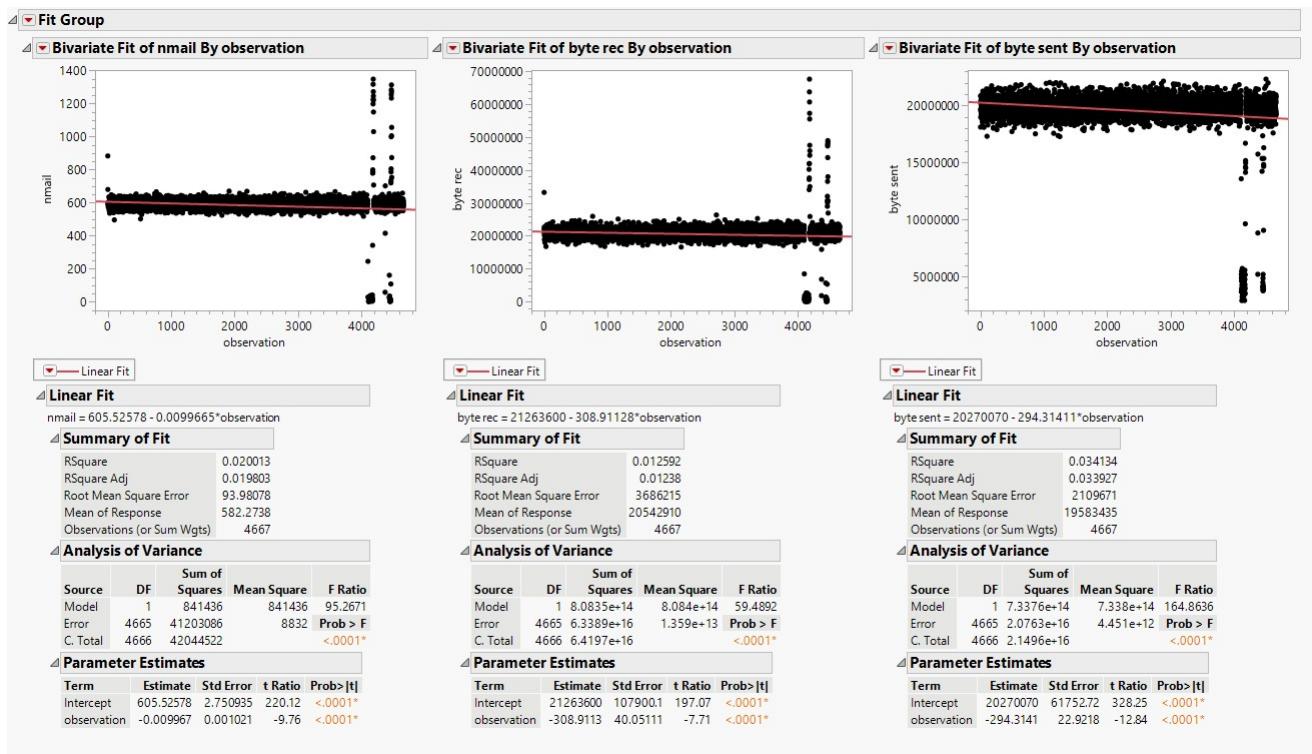


Figura 6.4: Stima lineare EXP2

Anche in questo caso l'R-quadro ha valori molto bassi, tuttavia lo scopo dell'homework è quello di fare trend detection e per questo motivo si possono prendere anche valori piccoli di R-quadro.

Successivamente occorre verificare la significatività dei parametri della retta regressiva, in particolare della **pendenza**, tramite un opportuno test statistico.

6.3.2 Verifica delle assunzioni

Per verificare le assunzioni del modello regressivo, si può utilizzare un test visivo per validare l'ipotesi di **omoschedasticità** e **normalità** dei **residui**. Le ipotesi implicano che le differenze tra i valori previsti dal modello e i valori osservati abbiano una varianza costante per ogni valore osservato e che seguano una distribuzione normale.

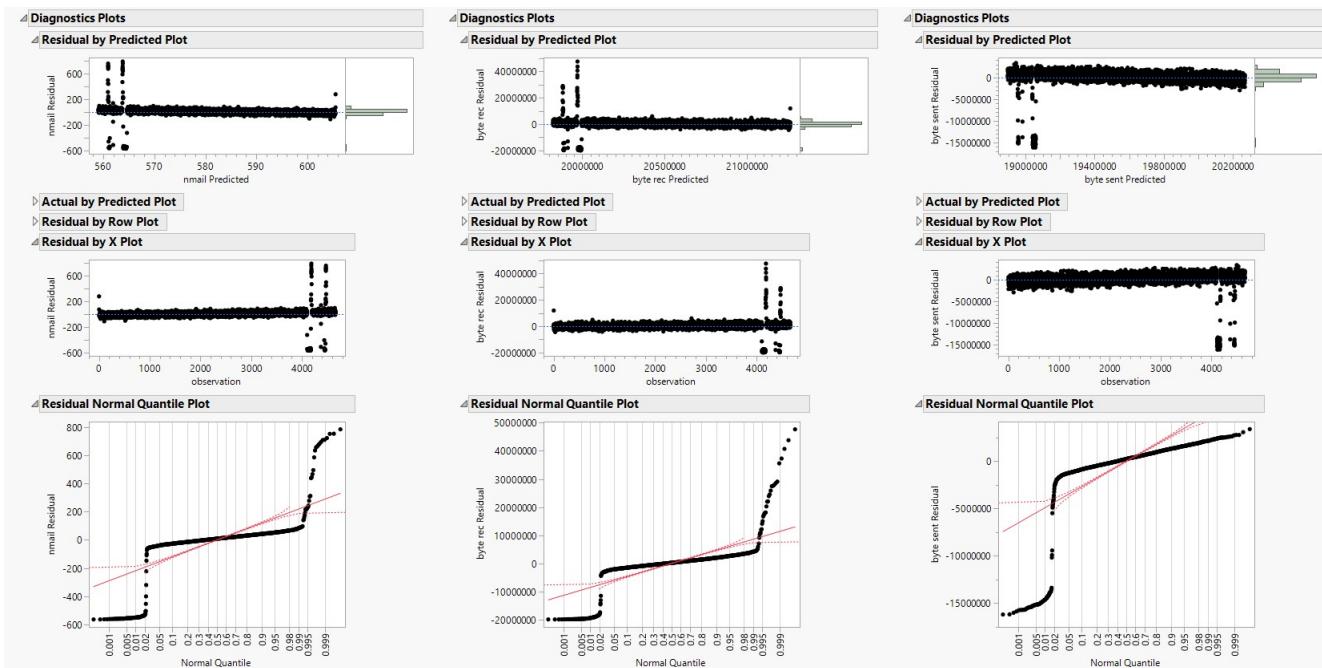


Figura 6.5: Verifica assunzioni EXP2

Come si può notare dai test visivi, l'ipotesi di normalità non può essere soddisfatta poiché i dati non seguono la curva di confidenza della normale. Di conseguenza si applica un test **non parametrico** ossia il test di **Mann-Kendall**.

6.3.3 Stima dei parametri regressivi

Tramite JMP si può calcolare il coefficiente τ di **Kendall** e il p-value associato al test. I valori di τ vanno da -1 a 1; se il valore è negativo indica che le variabili in esame sono inversamente correlate altrimenti sono direttamente correlate, ossia c'è una diretta proporzionalità. L'ipotesi nulla del test H₀ indica la **non** presenza di un trend monotono nella distribuzione, quindi se il p-value è inferiore o uguale a 0.05 significa che il risultato è significativo.

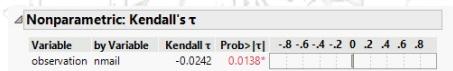


Figura 6.6: Kendall EXP2 - nmail

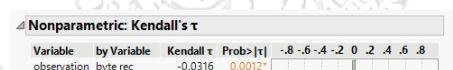


Figura 6.7: Kendall EXP2 - byte rec

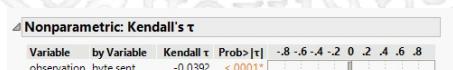


Figura 6.8: Kendall EXP2- byte sent

Dalle tabelle si può vedere che è presente un trend per quanto riguarda le tre variabili in esame. Infine, si va a valutare la pendenza della retta di regressione utilizzando la **procedura di Theil-Sen**, ossia una procedura lineare robusta che fornisce risultati precisi in presenza di valori outlier nei dati. Per il calcolo si utilizza uno script python con la funzione theilslope fornita dalla libreria SciPy.

	nmail	byte rec	byte sent
Pendenza	-0.0005973715651135006	-48.29360967184802	-34.06305208650625
CI inferiore	-0.0011248593925759281	-77.54773269689737	-50.672672672672675
CI superiore	0.0	-18.993146773272414	-17.374087591240876

Tabella 6.1: Theil-Sen EXP2

Nel caso di nmail l'intervallo di confidenza al 95% relativo alla pendenza della retta di regressione include lo zero, dunque la pendenza relativa al modello della popolazione ha una probabilità del 95% che sia significativamente uguale a zero. Per gli altri due casi di conseguenza si può dire che c'è un trend significativo.

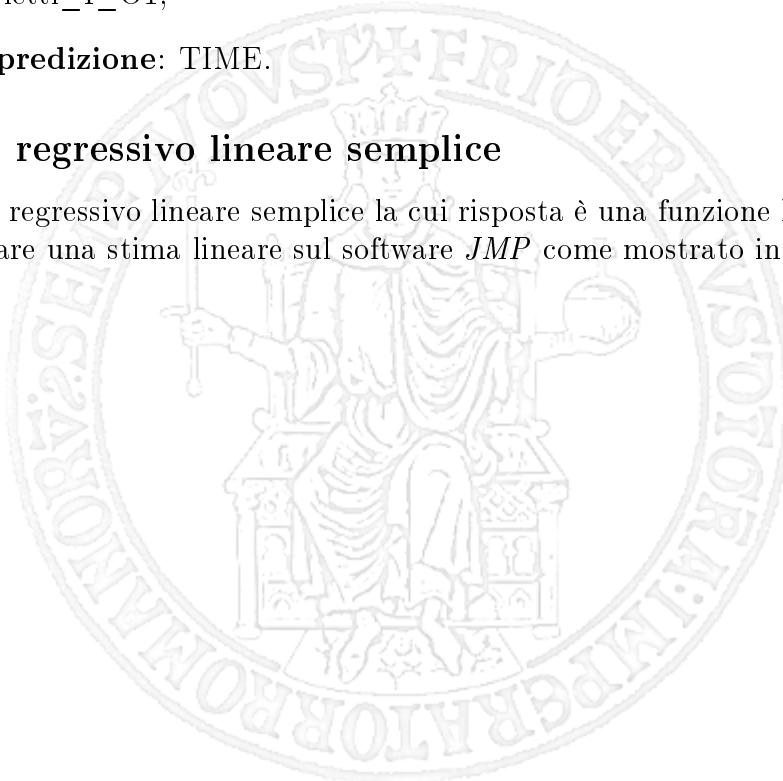
6.4 Dataset os1

Il dataset è formato da:

- **Variabili di risposta:** LIN1_VmSize, LIN1_VmData, LIN1_RSS, LIN1_byte_letti_I_O e LIN1_byte_letti_I_O1;
- **Variabile di predizione:** TIME.

6.4.1 Modello regressivo lineare semplice

Si applica il modello regressivo lineare semplice la cui risposta è una funzione lineare di un singolo predittore. Si va a fare una stima lineare sul software *JMP* come mostrato in figura.



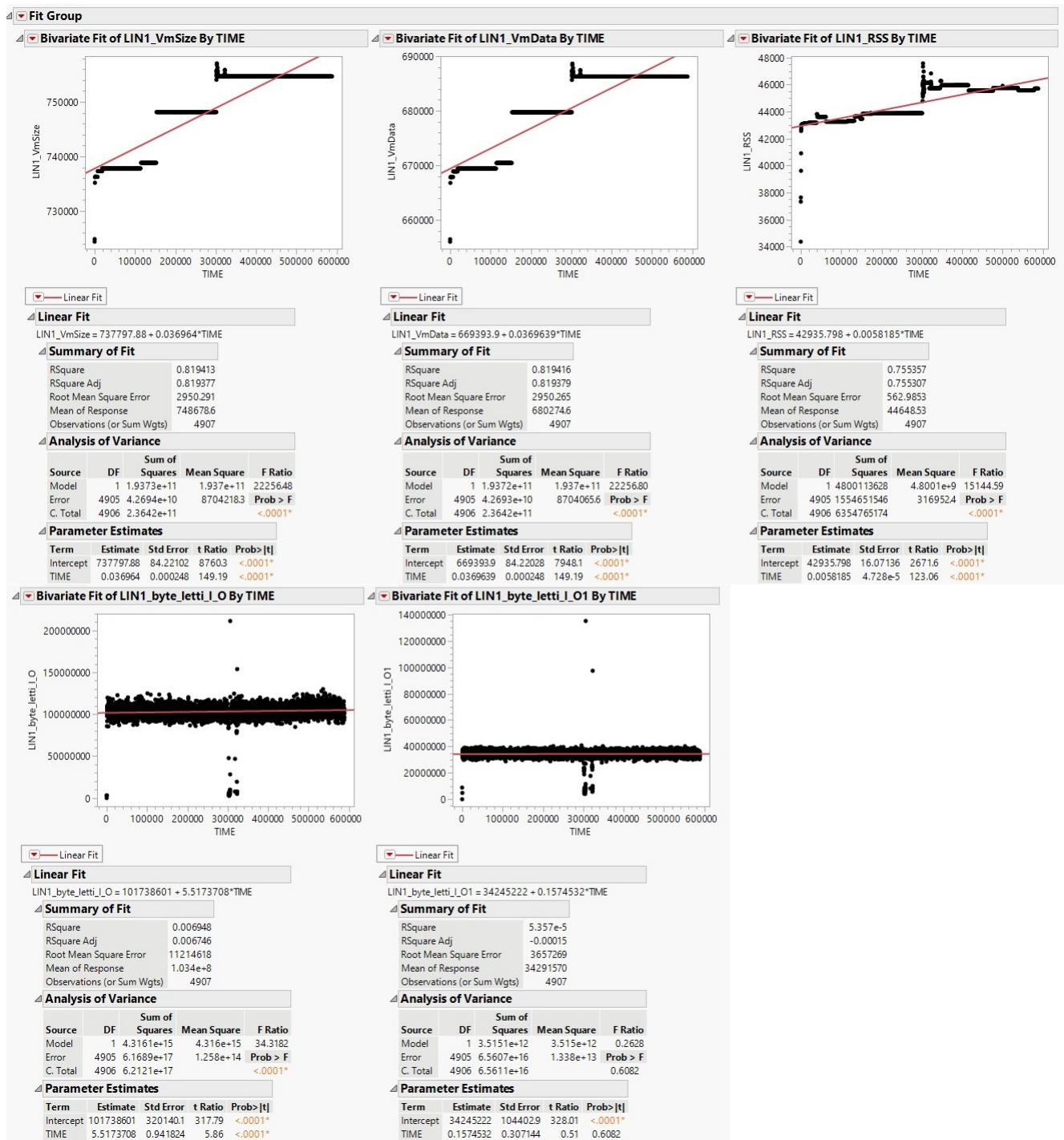


Figura 6.9: Stima lineare os1

In questo caso l'R-quadro ha anche valori più alti, tranne per gli ultimi due casi, tuttavia lo scopo dell'homework è quello di fare trend detection e per questo motivo si possono prendere anche valori piccoli di R-quadro.

Successivamente occorre verificare la significatività dei parametri della retta regressiva, in particolare della **pendenza**, tramite un opportuno test statistico.

6.4.2 Verifica delle assunzioni

Per verificare le assunzioni del modello regressivo, si può utilizzare un test visivo per validare l'ipotesi di **omoschedasticità** e **normalità** dei **residui**. Le ipotesi implicano che le differenze tra i valori previsti dal modello e i valori osservati abbiano una varianza costante per ogni valore osservato e che seguano una distribuzione normale.

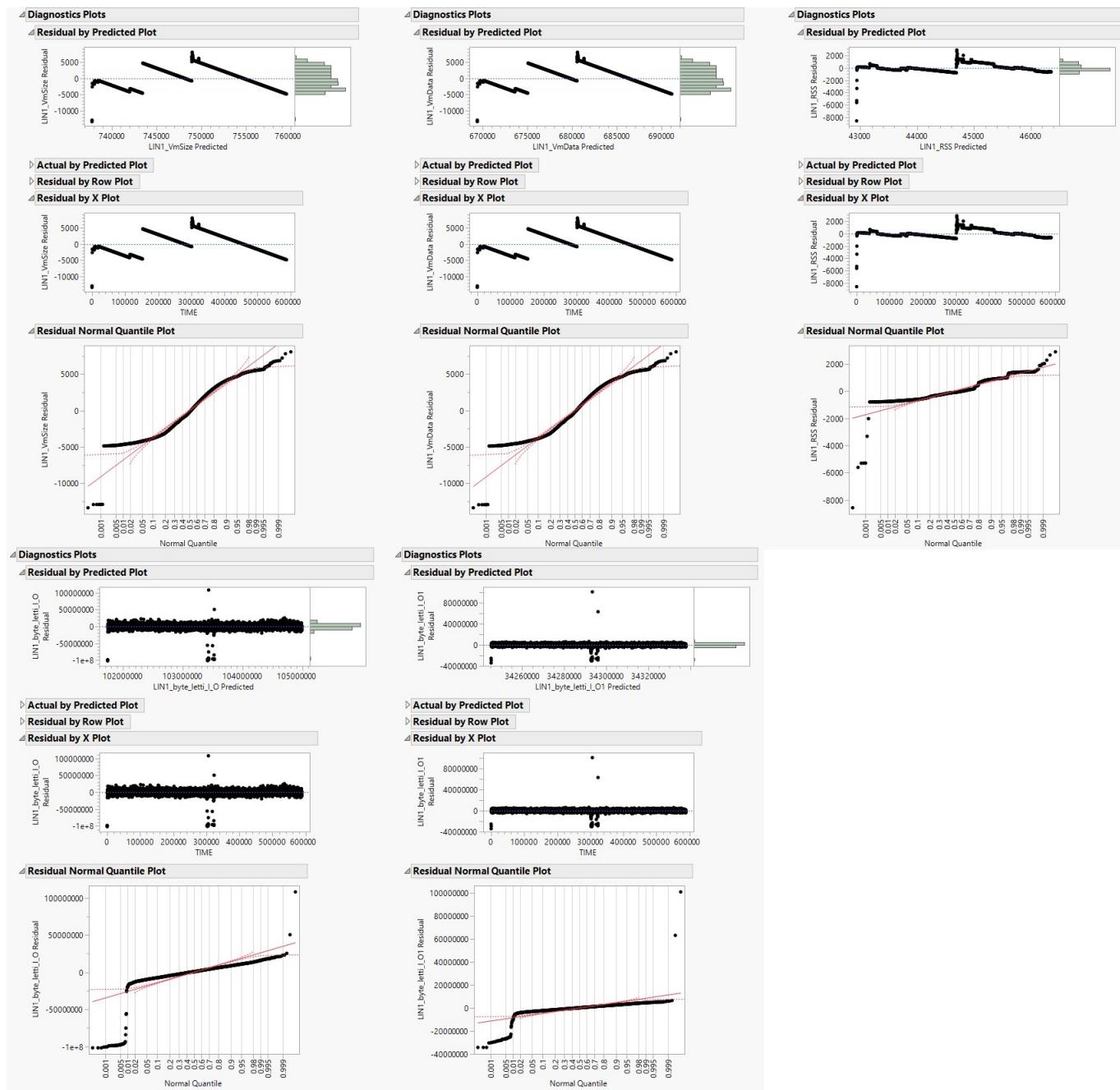


Figura 6.10: Verifica assunzioni os1

Come si può notare dai test visivi, le ipotesi di omoschedasticità e normalità non sono soddisfatte per le prime tre variabili. Tuttavia, si possono notare degli outliers nei casi di

LIN1_byte_letti_I_O e LIN1_byte_letti_I_O1. Si conclude che i valori sono outliers per entrambe le variabili e quindi si decide di filtrarli.

A valle del filtraggio, le assunzioni possono essere soddisfatte per LIN1_byte_letti_I_O e LIN1_byte_letti_I_O1 dunque si applica il test parametrico (test sulla media) e il test non parametrico di Mann-Kendall per le altre tre variabili.

6.4.3 Stima dei parametri regressivi

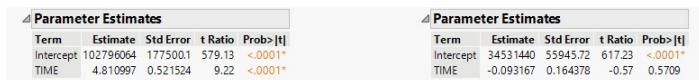


Figura 6.11: Stima parametri os1 - byte_letti_I_O e byte_letti_I_O1

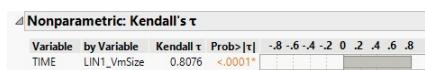


Figura 6.12: Kendall os1 - VmSize

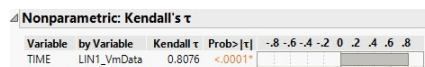


Figura 6.13: Kendall os1 - VmData



Figura 6.14: Kendall os1 - RSS

Si va a valutare anche in questo caso la pendenza della retta di regressione utilizzando la **procedura di Theil-Sen**.

	VmSize	VmData	RSS
Pendenza	0.03157224220623501	0.03157224220623501	0.004970760233918129
CI inferiore	0.030715952172645087	0.030715952172645087	0.004876649454962708
CI superiore	0.03232097662647015	0.03232097662647015	0.005055788005578801

Tabella 6.2: Theil-Sen os1

Dalle tabelle si può notare che per tutte la variabile byte_letti_I_O1 l'intervallo di confidenza al 95% relativo alla pendenza della retta di regressione include lo zero dunque possiamo affermare che la pendenza, relativa al **modello della popolazione** ha una probabilità al 95% che sia significativamente uguale da zero. Per tutte le altre invece possiamo dire che è significativamente diversa da zero.

6.5 Dataset os2

Il dataset è formato da:

- **Variabili di risposta:** LIN2_VmSize, LIN2_VmData, LIN2_RSS, LIN2_byte_letti__sec e LIN2_byte_scritti__sec;
- **Variabile di predizione:** TIME.

6.5.1 Modello regressivo lineare semplice

Si applica il modello regressivo lineare semplice la cui risposta è una funzione lineare di un singolo predittore. Si va a fare una stima lineare sul software *JMP* come mostrato in figura.



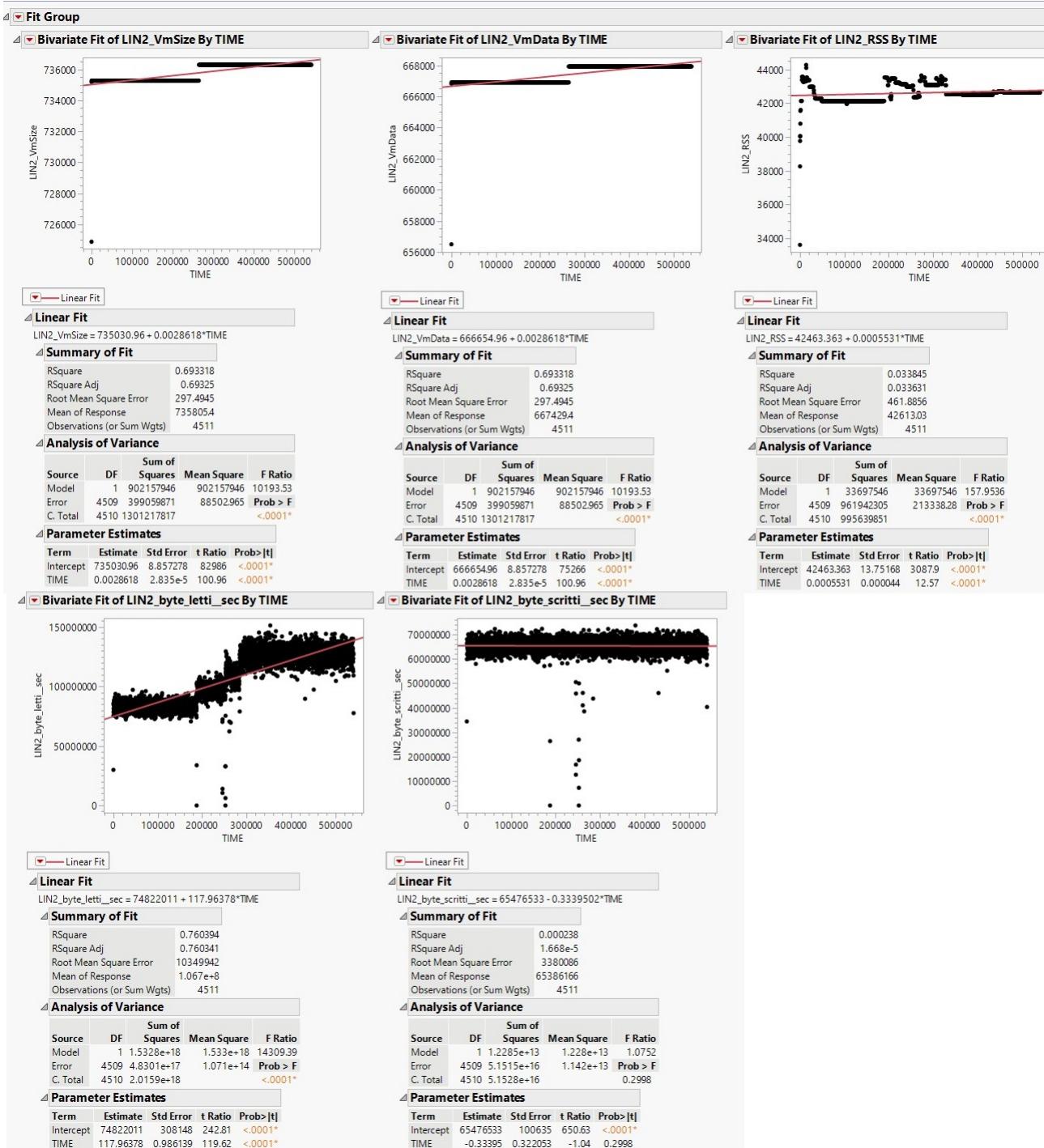


Figura 6.15: Stima lineare os2

In questo caso l'R-quadro ha valori diversi per ogni variabile, tuttavia lo scopo dell'homework è quello di fare trend detection e per questo motivo si possono prendere anche valori piccoli di R-quadro.

Successivamente occorre verificare la significatività dei parametri della retta regressiva, in particolare della **pendenza**, tramite un opportuno test statistico.

6.5.2 Verifica delle assunzioni

Per verificare le assunzioni del modello regressivo, si può utilizzare un test visivo per validare l'ipotesi di **omoschedasticità** e **normalità** dei **residui**. Le ipotesi implicano che le differenze tra i valori previsti dal modello e i valori osservati abbiano una varianza costante per ogni valore osservato e che seguano una distribuzione normale.

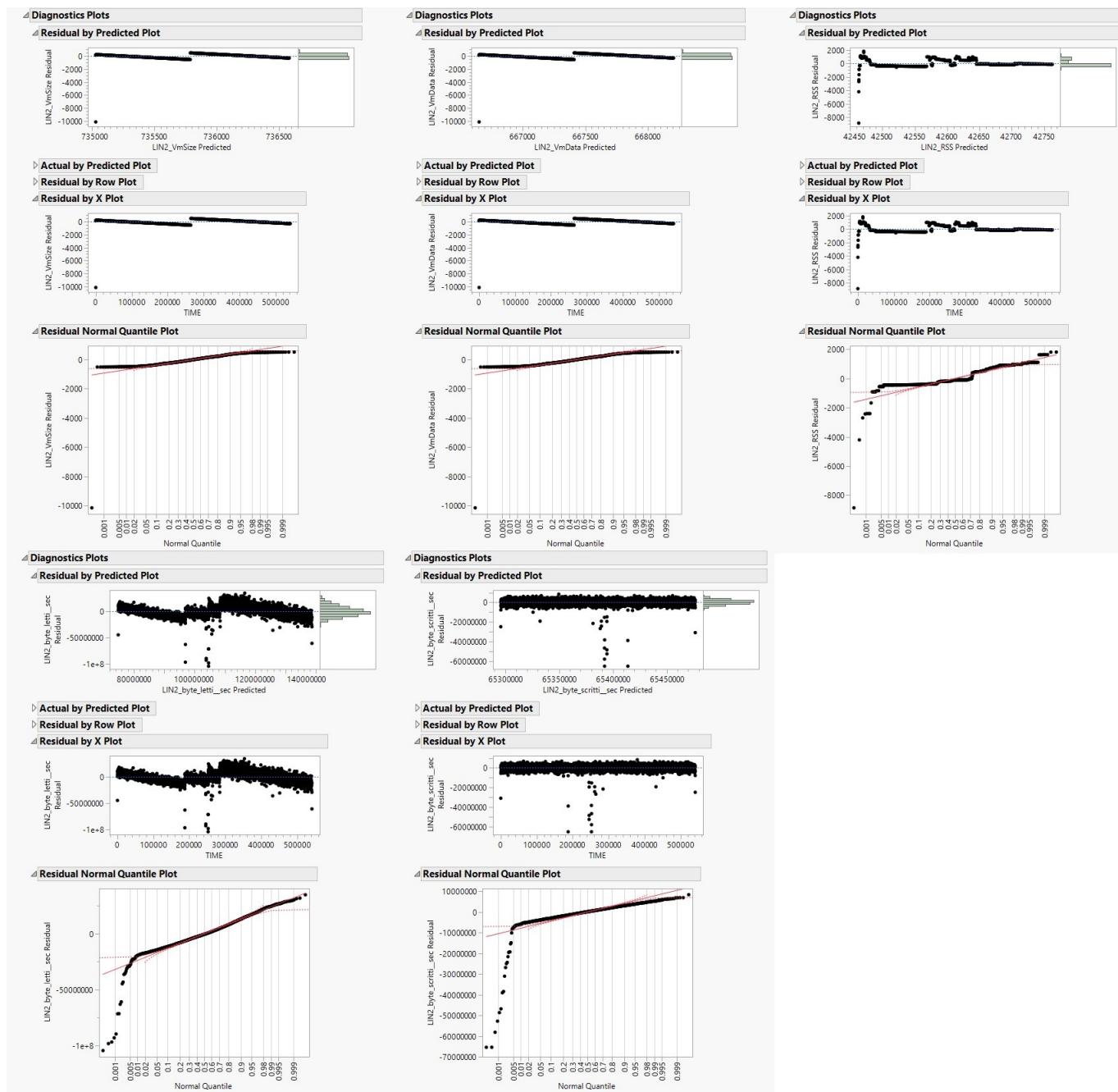


Figura 6.16: Verifica assunzioni os2

Come si può notare dai test visivi, le ipotesi di omoschedasticità e normalità non sono soddisfatte per le prime tre variabili. Tuttavia, anche in questo caso, si possono notare degli outliers nei casi di

`LIN2_byte_letti_sec` e `LIN2_byte_scritti_sec`. Si conclude che i valori sono outliers per le due variabili, e quindi, di conseguenza si decide di filtrarli. A valle del filtraggio, le assunzioni possono essere soddisfatte per `LIN2_byte_letti_sec` e `LIN1_byte_scritti_sec` dunque si applica il test parametrico (test sulla media) e il test non parametrico di Mann-Kendall per le altre tre variabili.

6.5.3 Stima dei parametri regressivi

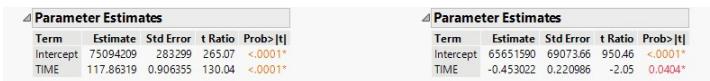


Figura 6.17: Stima parametri `os2 - byte_letti_sec` e `byte_scritti_sec`



Figura 6.18: Kendall os2 - VmSize

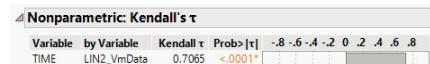


Figura 6.19: Kendall os2 - VmData

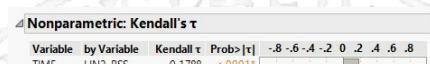


Figura 6.20: Kendall os2 - RSS

Si va a valutare anche in questo caso la pendenza della retta di regressione utilizzando la **procedura di Theil-Sen**.

	VmSize	VmData	RSS
Pendenza	0.0019128745423298214	0.0019128745423298214	0.000632466619817287
CI inferiore	0.0	0.0	0.00055555555555555556
CI superiore	0.002085369827305311	0.002085369827305311	0.0007445442875481386

Tabella 6.3: Theil-Sen os2

Dalle tabelle si può notare che per le variabili `byte_letti_sec`, `byte_scritti_sec` e `RSS` l'intervallo di confidenza al 95% relativo alla pendenza della retta di regressione non include lo zero, dunque possiamo affermare che la pendenza relativa al **modello della popolazione** ha una probabilità al 95% che sia significativamente diversa da zero. Per le altre due variabili, invece, si può dire che è significativamente uguale a zero.

6.6 Dataset os3

Il dataset è formato da:

- **Variabili di risposta:** LIN4_VmSize, LIN4_VmData, LIN4_RSS, LIN4_byte_letti__sec e LIN4_byte_scritti__sec;
- **Variabile di predizione:** TIME.

6.6.1 Modello regressivo lineare semplice

Si applica il modello regressivo lineare semplice la cui risposta è una funzione lineare di un singolo predittore. Si va a fare una stima lineare sul software *JMP* come mostrato in figura.



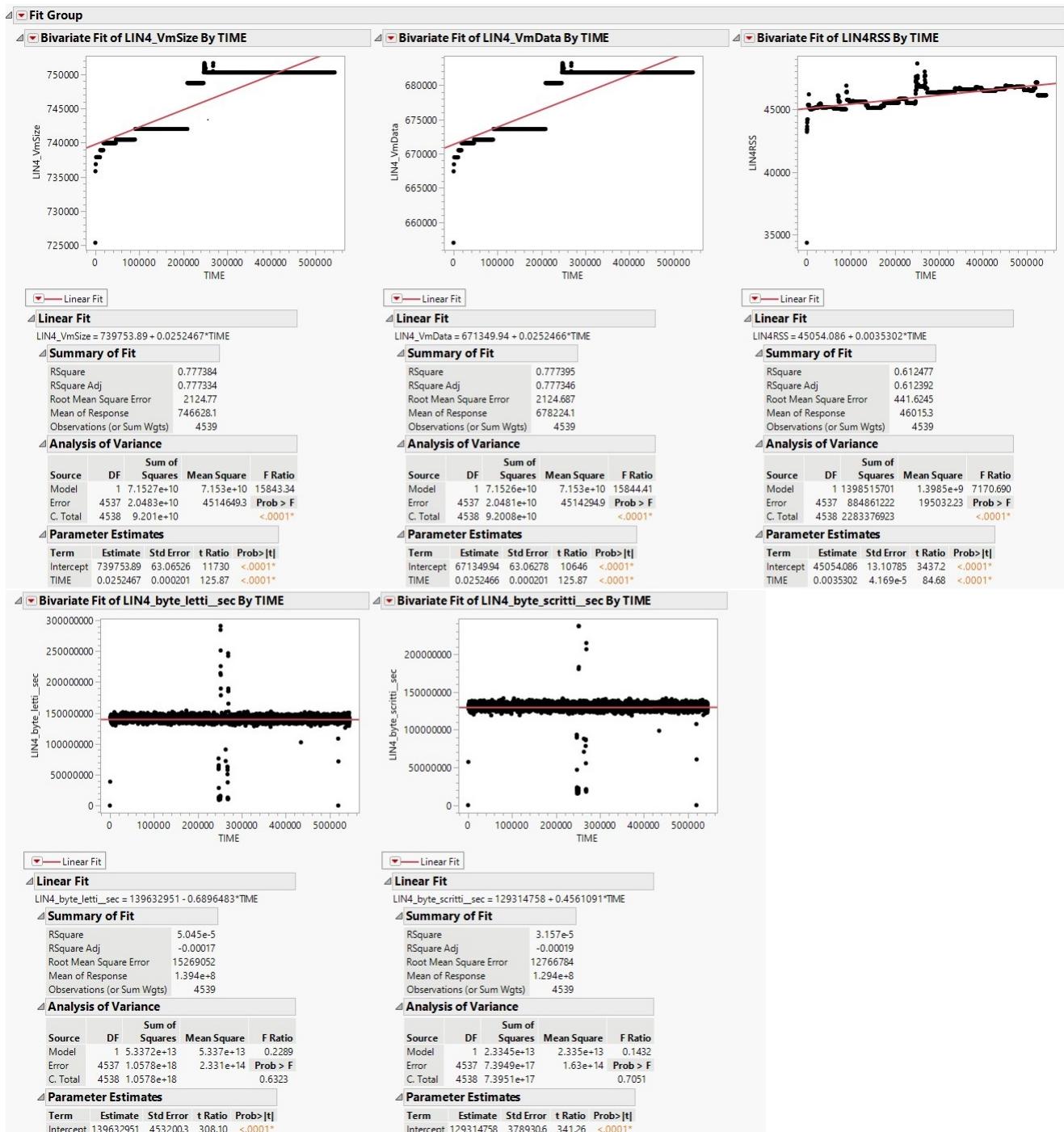


Figura 6.21: Stima lineare os3

In questo caso l'R-quadro ha valori più alti, tranne per gli ultimi due casi, tuttavia lo scopo dell'homework è quello di fare trend detection e per questo motivo si possono prendere anche valori piccoli di R-quadro.

Successivamente occorre verificare la significatività dei parametri della retta regressiva, in particolare della **pendenza**, tramite un opportuno test statistico.

6.6.2 Verifica delle assunzioni

Per verificare le assunzioni del modello regressivo, si può utilizzare un test visivo per validare l'ipotesi di **omoschedasticità** e **normalità** dei **residui**. Le ipotesi implicano che le differenze tra i valori previsti dal modello e i valori osservati abbiano una varianza costante per ogni valore osservato e che seguano una distribuzione normale.

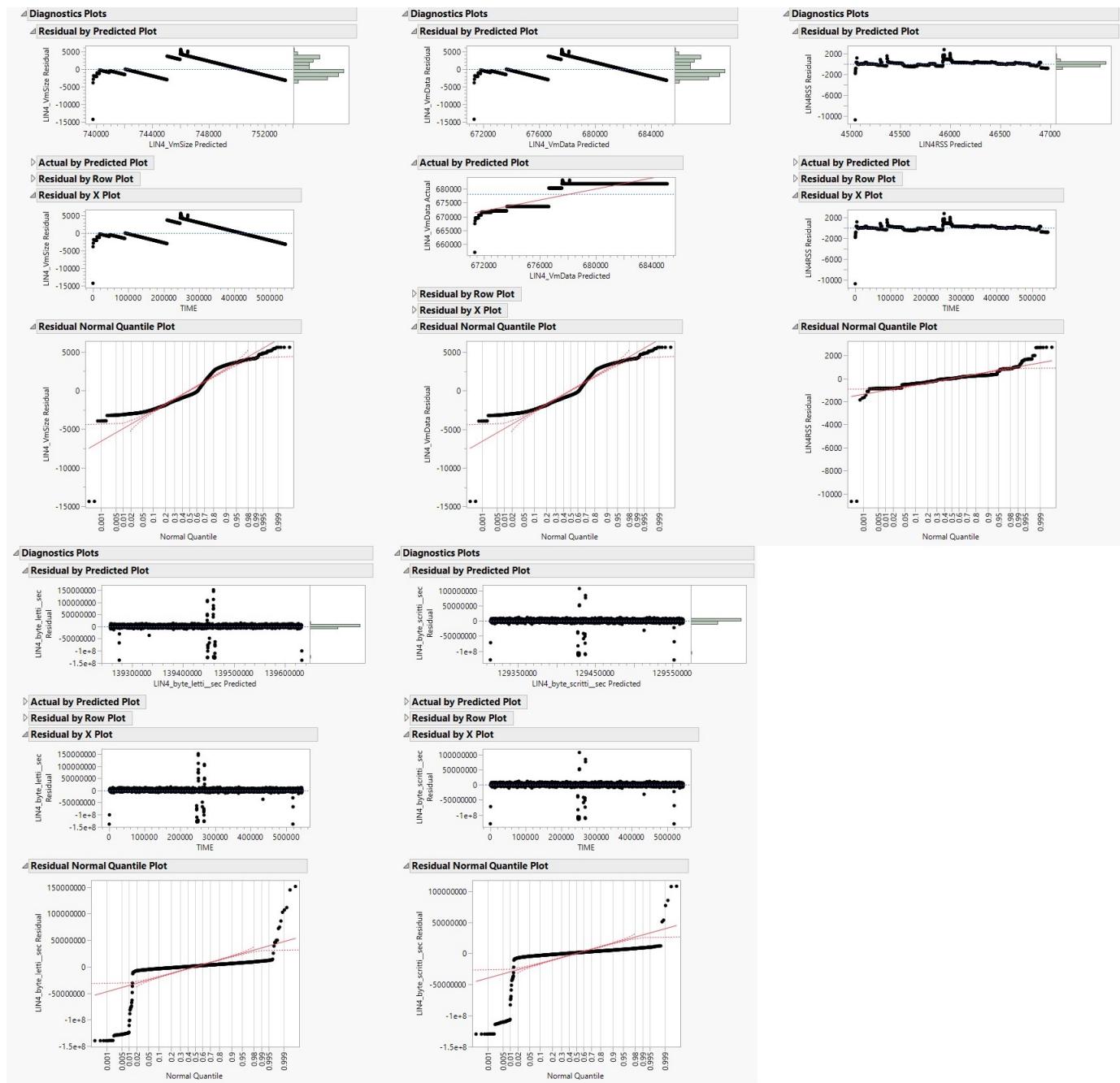


Figura 6.22: Verifica assunzioni os3

Anche in questo caso, come si può notare dai test visivi, le ipotesi di omoschedasticità e normalità non sono soddisfatte per le prime tre variabili. Tuttavia, si possono notare anche qui degli outliers.

nei casi di LIN4_byte_letti_sec e LIN4_byte_scritti_sec. Si conclude che i valori sono outliers per entrambe le variabili e quindi si decide di filtrarli.

A valle del filtraggio, le assunzioni possono essere soddisfatte per LIN4_byte_letti_sec e LIN4_byte_scritti_sec dunque si applica il test parametrico (test sulla media) e il test non parametrico di Mann-Kendall per le altre tre variabili.

6.6.3 Stima dei parametri regressivi

Parameter Estimates					
Term	Estimate	Std Error	t Ratio	Prob> t	
Intercept	141399133	1143465	1236.6	<.0001*	
TIME	-2.101592	0.362895	-5.79	<.0001*	

Parameter Estimates					
Term	Estimate	Std Error	t Ratio	Prob> t	
Intercept	130880931	9913949	1320.2	<.0001*	
TIME	-0.746715	0.314633	-2.37	0.0177*	

Figura 6.23: Stima parametri os3 - byte_letti_sec e byte_scritti_sec



Figura 6.24: Kendall os3 - VmSize



Figura 6.25: Kendall os3 - VmData

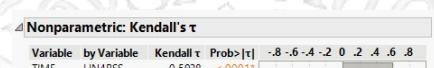


Figura 6.26: Kendall os3 - RSS

Si va a valutare anche in questo caso la pendenza della retta di regressione utilizzando la **procedura di Theil-Sen**.

	VmSize	VmData	RSS
Pendenza	0.021659324522760644	0.021659324522760644	0.0034579439252336447
CI inferiore	0.02127364220024293	0.02127364220024293	0.0033883947479881405
CI superiore	0.0220266666666666667	0.0220266666666666667	0.0035223947783029484

Tabella 6.4: Theil-Sen os3

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

6.7 Dataset VMres1

Il dataset è formato da:

- **Variabili di risposta:** Allocated Heap;
- **Variabile di predizione:** T(s).

6.7.1 Modello regressivo lineare semplice

Si applica il modello regressivo lineare semplice la cui risposta è una funzione lineare di un singolo predittore. Si va a fare una stima lineare sul software *JMP* come mostrato in figura.

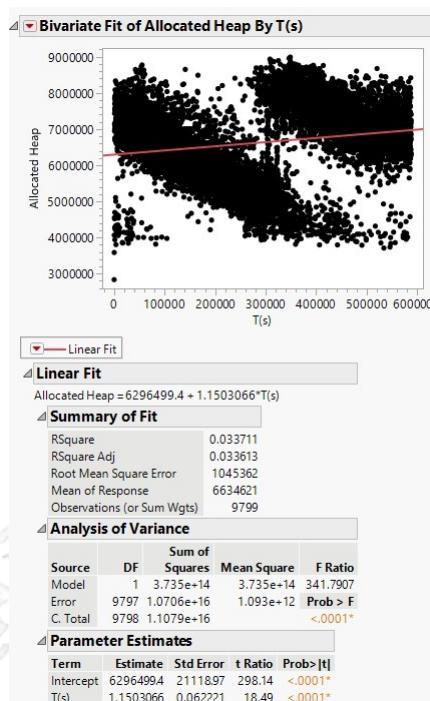


Figura 6.27: Stima lineare VMres1

In questo caso l'R-quadro ha valori bassi, tuttavia lo scopo dell'homework è quello di fare trend detection e per questo motivo si possono prendere anche valori piccoli di R-quadro.

Successivamente occorre verificare la significatività dei parametri della retta regressiva, in particolare della **pendenza**, tramite un opportuno test statistico.

6.7.2 Verifica delle assunzioni

Per verificare le assunzioni del modello regressivo, si può utilizzare un test visivo per validare l'ipotesi di **omoschedasticità** e **normalità** dei **residui**. Le ipotesi implicano che le differenze tra i valori previsti dal modello e i valori osservati abbiano una varianza costante per ogni valore osservato e che seguano una distribuzione normale.

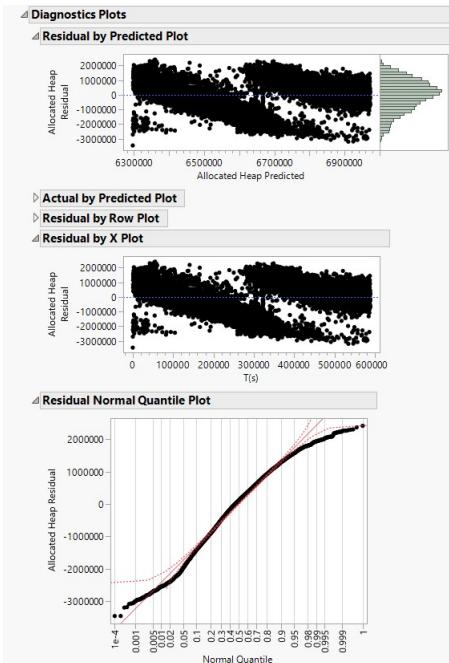


Figura 6.28: Verifica assunzioni VMres1

Come si può notare dai test visivi, le ipotesi di omoschedasticità e normalità possono essere soddisfatte nonostante la presenza di outliers che però vengono ritenuti ininfluenti per lo scopo dell'analisi.

Dunque si applica il test parametrico (test sulla media) per stimare i parametri regressivi.

6.7.3 Stima dei parametri regressivi

Parameter Estimates						
Term	Estimate	Std Error	t Ratio	Prob> t	Lower 95%	Upper 95%
Intercept	6296499.4	2111897	298.14	<.0001*	6255101.9	6327897
T(s)	1.1503066	0.062221	18.49	<.0001*	1.0283416	1.2722716

Figura 6.29: Stima parametri VMres1

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

6.7.4 Failure prediction

Il tempo di fallimento si può ricavare dalla retta di regressione:

$$\text{AllocatedHeap} = b_0 + b_1 T(s)$$

Sostituendo i valori della pendenza e dell'intercetta trovati precedentemente e sostituendo il limite massimo di Allocated Heap con 1 GB (1073741824 byte) si può ricercare il tempo in cui l'heap satura.

$$T(s) = \frac{(1073741824 - (6296499.4 \pm 41397.5))}{(1.15 \pm 0.125)} \simeq 30 \pm 3 \text{ anni}$$

6.8 Dataset VMres2

Il dataset è formato da:

- **Variabili di risposta:** Allocated Heap;
- **Variabile di predizione:** T(s).

6.8.1 Modello regressivo lineare semplice

Si applica il modello regressivo lineare semplice la cui risposta è una funzione lineare di un singolo predittore. Si va a fare una stima lineare sul software *JMP* come mostrato in figura.

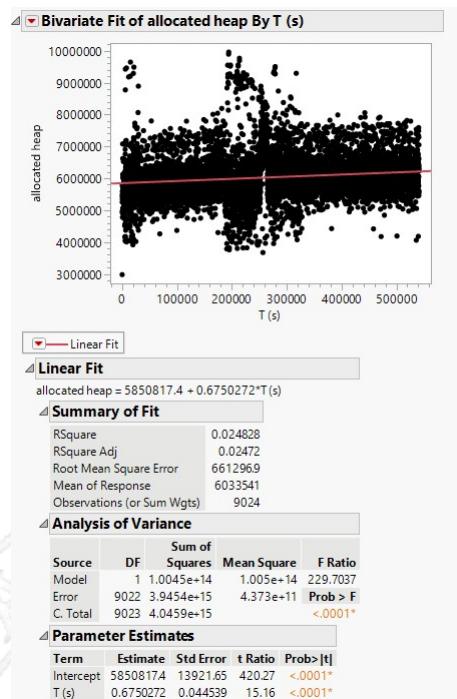


Figura 6.30: Stima lineare VMres2

In questo caso l'R-quadro ha valori bassi, tuttavia lo scopo dell'homework è quello di fare trend detection e per questo motivo si possono prendere anche valori piccoli di R-quadro.

Successivamente occorre verificare la significatività dei parametri della retta regressiva, in particolare della **pendenza**, tramite un opportuno test statistico.

6.8.2 Verifica delle assunzioni

Per verificare le assunzioni del modello regressivo, si può utilizzare un test visivo per validare l'ipotesi di **omoschedasticità** e **normalità** dei **residui**. Le ipotesi implicano che le differenze tra i valori previsti dal modello e i valori osservati abbiano una varianza costante per ogni valore osservato e che seguano una distribuzione normale.

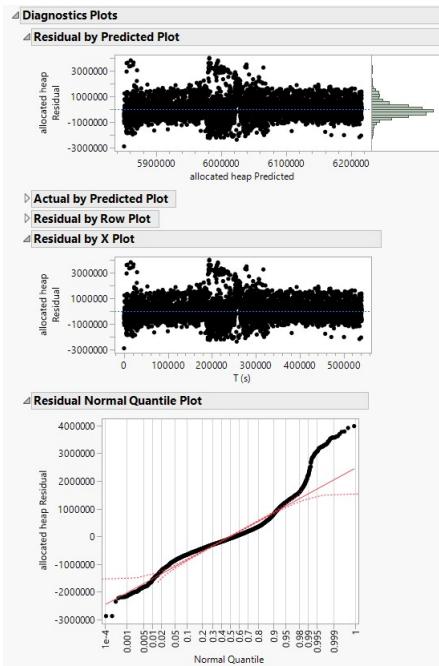


Figura 6.31: Verifica assunzioni VMres2

Come si può notare dai test visivi, anche in questo caso, le ipotesi di omoschedasticità e normalità possono essere soddisfatte nonostante la presenza di outliers che però vengono ritenuti ininfluenti per lo scopo dell'analisi.

Dunque si applica il test parametrico (test sulla media) per stimare i parametri regressivi.

6.8.3 Stima dei parametri regressivi

Parameter Estimates						
Term	Estimate	Std Error	t Ratio	Prob> t	Lower 95%	Upper 95%
Intercept	58508174	13921.65	420.27	<.0001*	5823527.8	5878107
T (s)	0.6750272	0.044539	15.16	<.0001*	0.5877213	0.7623331

Figura 6.32: Stima parametri VMres2

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

6.8.4 Failure prediction

Il tempo di fallimento si può ricavare dalla retta di regressione:

$$\text{AllocatedHeap} = b_0 + b_1 T(s)$$

Sostituendo i valori della pendenza e dell'intercetta trovati precedentemente e sostituendo il limite massimo di Allocated Heap con 1 GB (1073741824 byte) si può ricercare il tempo in cui l'heap satura.

$$T(s) = \frac{(1073741824 - (5823527.8 \pm 27289.6))}{(0.67 \pm 0.09)} \simeq 51 \pm 7 \text{ anni}$$

6.9 Dataset VMres3

Il dataset è formato da:

- **Variabili di risposta:** Allocated Heap;
- **Variabile di predizione:** T(s).

6.9.1 Modello regressivo lineare semplice

Si applica il modello regressivo lineare semplice la cui risposta è una funzione lineare di un singolo predittore. Si va a fare una stima lineare sul software *JMP* come mostrato in figura.

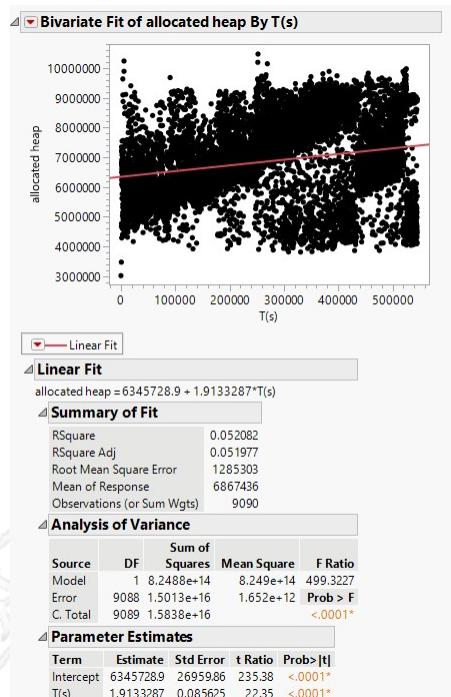


Figura 6.33: Stima lineare VMres3

In questo caso l'R-quadro ha valori bassi, tuttavia lo scopo dell'homework è quello di fare trend detection e per questo motivo si possono prendere anche valori piccoli di R-quadro.

Successivamente occorre verificare la significatività dei parametri della retta regressiva, in particolare della **pendenza**, tramite un opportuno test statistico.

6.9.2 Verifica delle assunzioni

Per verificare le assunzioni del modello regressivo, si può utilizzare un test visivo per validare l'ipotesi di **omoschedasticità** e **normalità** dei **residui**. Le ipotesi implicano che le differenze tra i valori previsti dal modello e i valori osservati abbiano una varianza costante per ogni valore osservato e che seguano una distribuzione normale.

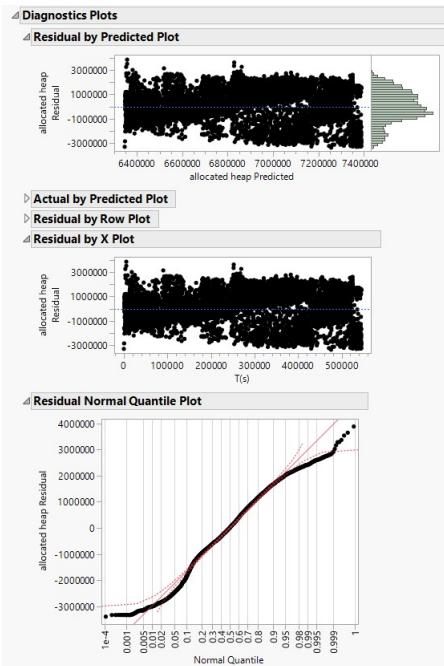


Figura 6.34: Verifica assunzioni VMres3

Come si può notare dai test visivi, anche in questo caso, le ipotesi di omoschedasticità e normalità possono essere soddisfatte nonostante la presenza di outliers che però vengono ritenuti ininfluenti per lo scopo dell'analisi.

Dunque si applica il test parametrico (test sulla media) per stimare i parametri regressivi.

6.9.3 Stima dei parametri regressivi

Parameter Estimates						
Term	Estimate	Std Error	t Ratio	Prob> t	Lower 95%	Upper 95%
Intercept	6345728.9	2695986	235.38	<.0001*	6292881.5	6398576.3
T(s)	1.9133287	0.085625	22.35	<.0001*	1.7454851	2.0811723

Figura 6.35: Stima parametri VMres2

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

6.9.4 Failure prediction

Il tempo di fallimento si può ricavare dalla retta di regressione:

$$\text{AllocatedHeap} = b_0 + b_1 T(s)$$

Sostituendo i valori della pendenza e dell'intercetta trovati precedentemente e sostituendo il limite massimo di Allocated Heap con 1 GB (1073741824 byte) si può ricercare il tempo in cui l'heap satura.

$$T(s) = \frac{(1073741824 - (6345728.9 \pm 52847.4))}{(1.91 \pm 0.17)} \simeq 17 \pm 1 \text{ anni}$$