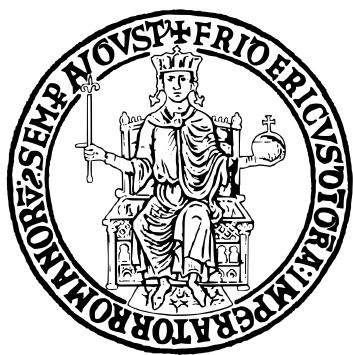


UNIVERSITA DEGLI STUDI DI NAPOLI
FEDERICO II



Corso di Laurea Magistrale in Ingegneria Informatica
Dipartimento di Ingegneria Elettrica e delle tecnologie
dell'informazione

Elaborato finale in
IMPIANTI DI ELABORAZIONE

Professore
Domenico Cotroneo

Studenti
Emanuele Riccio M63001339
Raimondo Reggio M63001328
Vincenzo Pascarella M63001337

Indice

1	Benchmark	1
1.1	Descrizione del problema	1
1.2	Raccolta dei dati	2
1.3	Confronto dei sistemi	4
2	Workload Characterization	6
2.1	Traccia	6
2.2	Workload Characterization	6
2.2.1	Preprocessing	7
2.2.2	Principal Component Analysis (PCA)	11
2.2.3	Clustering	13
2.2.4	Calcolo delle devianze	17
2.3	Analisi delle devianze	20
2.3.1	Devianza persa vs Numero di componenti principali	20
2.3.2	Devianza persa vs Dimensione del workload sintetico	22
2.3.3	Workload sintetico	23
3	Web Server	26
3.1	Capacity Test	26
3.1.1	Traccia	26

INDICE

3.1.2	Introduzione	27
3.1.3	Esecuzione del test	28
3.2	Workload characterization	31
3.2.1	Fase1	32
3.2.2	Fase2	42
3.2.3	Fase3	45
3.3	Design of Experiment	51
3.3.1	Effetti dei fattori	52
3.3.2	Errori	53
3.3.3	Importanza	54
3.3.4	Significatività	55
4	Regression Model	60
4.1	Descrizione del problema	60
4.2	Traccia 1	61
4.2.1	Exp1	61
4.2.2	Exp2	70
4.3	Traccia 2	77
4.3.1	Risultati os1, os2, os3	77
4.4	Traccia 3	81
4.4.1	Vmres1	82
4.4.2	Vmres2	83
4.4.3	Vmres3	85
5	RBD	88
5.1	Esercizio 1	88
5.1.1	Traccia	88
5.1.2	Calcolo della reliability	89

5.1.3	Calcolo del MTTF	91
5.2	Esercizio 2	92
5.2.1	Traccia	92
5.2.2	Svolgimento	93
5.3	Esercizio 3	97
5.3.1	Traccia	97
5.3.2	Svolgimento	98
5.4	Esercizio 4	100
5.4.1	Traccia	100
5.4.2	Svolgimento	100
5.5	Esercizio 5	104
5.5.1	Traccia	104
6	Field Failure Data Analysis	111
6.1	Traccia	111
6.2	Field Failure Data Analysis (FFDA)	112
6.3	Mercury	113
6.3.1	Descrizione del sistema	113
6.3.2	Fase di Data Manipulation del sistema	114
6.3.3	Fase di Data Analysis del sistema	120
6.3.4	Fase di Data Manipulation dei componenti	123
6.3.5	Fase di Data Analysis dei componenti	128
6.3.6	Fase di Data Manipulation dei nodi	131
6.3.7	Fase di Data Analysis dei nodi	136
6.3.8	Domande	141
6.4	Blue Gene/L	145
6.4.1	Descrizione del sistema	145
6.4.2	Fase di Data Manipulation del sistema	147

INDICE

6.4.3	Fase di Data Analysis del sistema	150
6.4.4	Fase di Data Manipulation dei nodi	152
6.4.5	Fase di Data Analysis dei rack	160
6.4.6	Fase di Data Manipulation delle card	166
6.4.7	Fase di Data Analysis delle card	169
6.4.8	Domande	171
6.4.9	Confronto tra Mercury e Blue Gene/L	172

Capitolo 1

Benchmark

1.1 Descrizione del problema

Al fine di confrontare le prestazioni di due sistemi, sono stati entrambi sottoposti al benchmark **Nbody**. Tale simulazione considera N corpi liberi di muoversi sotto l'azione della forza di attrazione gravitazionale reciproca. La meccanica classica permette di impostare un sistema di $3N$ equazioni differenziali ordinarie accoppiate di second'ordine che si possono ridurre a $3(N-1)$ nel sistema del centro di massa. Fissate le condizioni iniziali di ogni corpo, la sua posizione e la sua velocità nello specifico, resta definito un problema di Cauchy che può essere trattato matematicamente. Il problema matematico definito dal benchmark è utile in tale analisi perché le operazioni eseguite mettono sotto stress non solo le FLops (Floating Point operations), ma anche lo stack e il numero di core. Infatti, il **SUT** (System Under Test) è il processore ed il **CUS** (Component Under Study) sono le unità Floating Point. La valutazione che si vuole effettuare è relativa alle prestazioni del processore nell'eseguire operazioni floating point. In particolare il benchmark

restituisce un solo parametro di interesse, Time(ms) che corrisponde al tempo in microsecondi per costruire la simulazione. Il confronto è stato effettuato considerando le seguenti dimensioni: 50000, 100000, 500000, 1000000. Nel caso specifico, il benchmark è stato utilizzato per comparare i seguenti sistemi:

Processore	Intel Core i7-8565U	Rayzen 7 5000 series
Sistema Operativo	Windows 11	Windows 11
Cores	8	8
Frequenza base	1.8GHz	2.2GHz
Memoria Ram	8Gb	16Gb

Tabella 1.1: Caratteristiche sistemi

1.2 Raccolta dei dati

Per ogni dimensione, si avvia il PC esegue 5 volte Nbody, append dei 5 valori in NREP, riavvia, finchè il file non contiene 200 righe, quindi 40 volte

Per ogni dimensione fissata di corpi sono state raccolte 40 osservazioni su entrambi i sistemi, eseguendo un singolo test e riavviando ogni volta il sistema, ripetendo la raccolta 5 volte e mediando i risultati ottenuti. Tale procedura assicura che le osservazioni siano tra loro indipendenti e identicamente distribuite. In questo modo sono verificate le ipotesi del teorema del limite centrale, il quale garantisce che al tendere ad infinito della dimensione di un campione, la media campionaria tende ad una distribuzione normale. Di conseguenza, essendo valida l'ipotesi di normalità è possibile applicare test parametrici. Le osservazioni raccolte per ogni dimensione costituiscono il precampione, utilizzato per stimare la deviazione standard della popolazione. Ottenuta tale informazione è possibile calcolare la dimensione campionaria, ovvero il numero di esperimenti da fare per ottenere che, con un livello di confidenza del 95%, l'errore sulla stima della media della

il numero di precampioni che raccolgo è 40 per il TLC così posso approssimare la sigma della popolaz con quella del campione. La demsione campionaria è la n che calcolo dalla formula

popolazione non sia superiore a 5%. Essa è stata calcolata quindi come:

$$n = \left(\frac{z_\alpha * \sigma}{E} \right)^2 \quad \text{sigma} = S/\sqrt{n} \quad (1.1)$$

dove:

$$E = |x^* - \mu| \\ \text{lo settiamo } 0.05$$

- E è l'errore massimo commesso nella stima della media della popolazione
- z è uguale a 1.96 è l'alfa-quantile della distribuzione normale, con alpha pari a 0.95;

I risultati ottenuti sono:

	50000	100000	500000	1000000
Intel Core i7-8565U	12	13	3	1
Rayzen 7 5000 series	27	33	15	7

Tabella 1.2: Dimensione campionaria

Una volta determinata la dimensione campionaria opportuna, sono state eventualmente raccolte ulteriori osservazioni, in modo da ottenere un campione per ogni dimensione e per ogni sistema. I dati raccolti hanno evidenziato però una dimensione campionaria uguale o inferiore alla dimensione del precampione già raccolto, come mostrato dalle tabelle seguenti, e si è quindi ritenuto superfluo raccogliere osservazioni aggiuntive.

Di seguito è riportata la media delle 40 osservazioni per ogni dimensione:

Intel Core i7-8565U				Rayzen 7 5000 series			
50000	100000	500000	1000000	50000	100000	500000	1000000
90616	183970	840018	1549033	30409	64372	292994	576820

Tabella 1.3: Media osservazioni

per ogni
dimensione 200
dati, media a 5
a 5, ottengo 40
punti e poi
media sui 40
punti

1.3 Confronto dei sistemi

Prima di effettuare un confronto sui dati ottenuti, occorre verificare che i campioni provenienti dalle varie distribuzioni siano statisticamente diversi. Nel dettaglio, l'obiettivo è quello di dimostrare che le differenze di prestazioni dei due sistemi sono statisticamente significative e che non sono dovute ad effetti aleatori. Per fare ciò, si è scelto di eseguire un **Two-sample t-test** con lo scopo di **rigettare** l'**ipotesi nulla H_0 : medie μ uguali**. Di seguito sono riportati i risultati ottenuti:

The screenshot shows the MATLAB interface with the following components:

- Editor Window:** Displays the script `test_benchmark.m` containing MATLAB code for performing a two-sample t-test across four dimensions.
- Workspace:** Shows variables and their values:

Name	Value
d1	'NREP1000000.txt'
d2	'NREPR1000000.txt'
data1	200x1 double
data2	200x1 double
data_mean_1	1x40 double
data_mean_2	1x40 double
h	1
h_t	1
h_value	[1;1;1]
i	4
p	2.9675e-58
p_t	1.1764e-38
p_value	[1.3372e-31;4.2763e-2...]
SIZE	[50000,100000,500000,...]
- Command Window:** Displays the output of the script execution, showing results for each dimension (i=1:4) indicating non-homogeneity and corresponding p-values.

```

test_benchmark.m × + | 
- clear all
- clc

- SIZE = [50000,100000,500000,1000000];

- h_value = zeros(4,1);
p_value = zeros(4,1);

- for i=1:4
    d1 = sprintf('NREP%d.txt',SIZE(i));
    d2 = sprintf('NREPR%d.txt',SIZE(i));

    data1 = importdata(d1);
    data2 = importdata(d2);

    data_mean_1 = arrayfun(@(k) mean(data1(k:min(k+5, length(data1))), 1:5:length(data1)), ...
    data_mean_2 = arrayfun(@(k) mean(data2(k:min(k+5, length(data2))), 1:5:length(data2)), ...

    [h,p]=vartest2(data_mean_1,data_mean_2);
    if h ==1
        fprintf('non-homo');
        [h_t,p_t]=ttest2(data_mean_1,data_mean_2,'vartype','unequal');
        disp(p_t);
        h_value(i) = h_t;
        p_value(i) = p_t;
    else
        fprintf('homo');
        [h_t,p_t]=ttest2(data_mean_1,data_mean_2);
        disp(p_t);
        h_value(i) = h_t;
        p_value(i) = p_t;
    end
end

immd Window
ew to MATLAB? See resources for Getting Started.
non-homo 1.3372e-31
non-homo 4.2763e-29
non-homo 5.0731e-35
non-homo 1.1764e-38

```

Figura 1.1: Analisi risultati

Da tali risultati si evince come l'**ipotesi nulla sia rigettata per tutte le dimensioni considerate**. Tale risultato ci porta a dire che i campioni raccolti dai due

CAPITOLO 1. BENCHMARK

sistemi sono statisticamente significativi. In generale però, come è possibile verificare dai dati ottenuti, il sistema con Rayzen 7 5000 series risulta mediamente più performante rispetto al sistema con Intel Core i7-8565U, perchè sono stati registrati tempi di esecuzione generalmente più bassi. Tali risultati confermano le specifiche dei due processori, in quanto:

- Il processore Intel Core i7-8565U ha una cache di livello 3 più grande rispetto al processore Ryzen 7 della serie 5000, il che può migliorare le prestazioni in applicazioni che richiedono l'accesso frequente alla cache. Inoltre, esso supporta la tecnologia Hyper-Threading, che permette di utilizzare più thread su ogni core, migliorando le prestazioni in alcune applicazioni multithreading. Tutte caratteristiche che non incidono sul nostro test.
- Il processore Ryzen 7 della serie 5000 ha una frequenza di clock più elevata rispetto al processore Intel Core i7-8565U, il che può migliorare le prestazioni in alcune applicazioni che richiedono calcoli veloci, come nel nostro caso.

Capitolo 2

Workload Characterization

2.1 Traccia

Effettua la caratterizzazione del workload su un sistema bancario, analizzando il dataset ed effettuando operazioni usando le tecniche di Principal Component Analysis e di Clustering. Successivamente studiare la devianza in funzione della dimensione del workload.

2.2 Workload Characterization

La **workload characterization** è il processo di creazione di un workload sintetico per il caso di test a partire da un workload reale. Si vuole ottenere una versione **sintetica del workload reale** che ne rappresenti la maggior parte della **variabilità**. Il workload da caratterizzare è costituito da 24 componenti (colonne) e 4995 osservazioni (righe), per un totale di 119.880 valori.

2.2.1 Preprocessing

E' necessario eseguire un'analisi preliminare del workload al fine di escludere componenti ed osservazioni non cruciali per la creazione del modello sintetico

Comprendere dei dati

Si è preferito andare a verificare cosa indica ogni attributo prima di usare approcci analitici.

- **VmPeak:** picco della memoria virtuale
- **VmSize:** dimensione della memoria virtuale
- **VmHWM:** picco della porzione di memoria virtuale occupata da un processo ("high water mark")
- **VmRSS:** dimensione della porzione di memoria virtuale occupata dal processo
- **VmPTE:** dimensione della tabella di paginazione della memoria virtuale
- **Threads:** Numero di thread nel processo che contiene questo thread
- **MemFree:** valore di memoria libera, derivante dalla somma di due quantità
- **Buffers:** archiviazione temporanea per blocchi di dischi grezzi che non dovrebbero diventare tremendamente grandi (20 MB circa)
- **Cached:** quantità di cache per la lettura di file da disco
- **Active:** memoria che è stata usata più recentemente e di solito non viene recuperata a meno che non sia assolutamente necessario

- **Inactive:** memoria che è stata usata meno di recente. È più idonea ad essere recuperata per altri scopi
- **Dirty:** memoria in attesa di essere riscritta
- **Writeback:** dati attivamente riscritti in memoria
- **AnonPages:** Pagine non supportate da file mappate in tabelle di pagina nello spazio utente
- **Mapped:** file mappati in memoria, come ad esempio librerie
- **Slab:** memoria usata dal kernel sotto forma di strutture dati cache per uso esclusivo
- **PageTables:** quantità di memoria dedicata al livello più basso di tabelle di pagina
- **Committed_AS:** quantità di memoria attualmente allocata sul sistema. La memoria impegnata è una somma di tutta la memoria che è stata allocata dai processi, anche se essi non l'hanno ancora "usata"
- **NumAllocFH:** descrizione non disponibile
- **proc-fd:** descrizione non disponibile
- **avgThroughput:** throughput medio applicato sul sistema
- **avgElapsed:** tempo medio di risposta delle richieste
- **avgLatency:** latenza media misurata
- **Errors:** numero di errori incorsi

Dati costanti

Si va a verificare la presenza di componenti(colonne) con valori tutti nulli, essendo essi distribuzioni a variabilità nulla non daranno nessun contributo alla varianza totale del workload. Nel caso trattato le colonne **AnonPages**, **avgLatency** e **Errors** risultano costanti, è quindi opportuno scartarle. Si passa da 24 a 21 componenti(colonne)

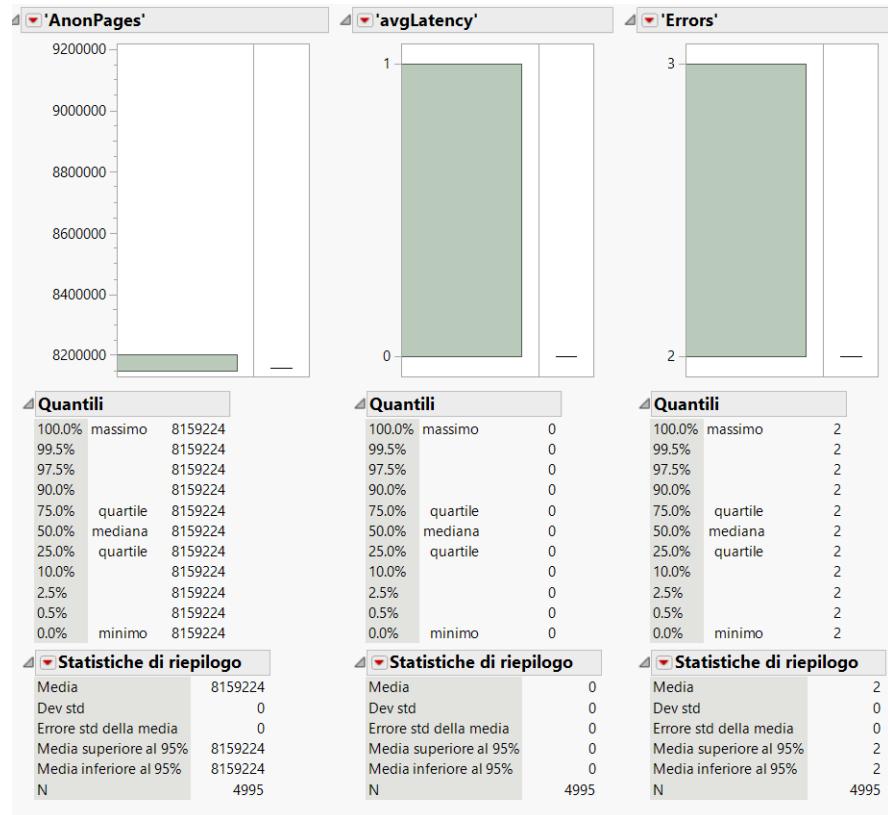


Figura 2.1: Colonne costanti

CAPITOLO 2. WORKLOAD CHARACTERIZATION

Dati perfettamente correlati

E' possibile che vi siano delle componenti perfettamente correlate, anche in questo caso tali componenti non contribuiscono alla varianza del workload. Si calcola la matrice di correlazione C degli componenti(colonne). L'elemento c_{ij} indica la correlazione tra gli componenti i e j, se si evidenzia un elemento della matrice al di fuori della diagonale pari a 1 significa che le due componenti i,j sono perfettamente correlate, quindi è possibile conservare solo una delle due.

Correlazioni	'VmPeak'	'VmSize'	'VmHWM'	'VmRSS'	'VmPTE'	'Threads'	'MemFree'	'Buffers'	'Cached'	'Active'	'Inactive'	'Dirty'	'WriteBack'	'Mapped'	'Slab'	'PageTables'	'CommittedAS'	'NumOfAllocFH'	'proc-fd'	'avgThroughput'	'avgElapsed'
'VmPeak'	1.0000	0.9900	0.9855	0.9389	0.9910	0.6312	-0.4668	0.6530	0.3069	-0.0012	0.7396	0.2508	-0.4669	-0.0605	-0.0113	0.8858	0.0696	0.4055	0.9650	-0.4491	
'VmSize'	0.9900	1.0000	0.9799	0.9628	0.9906	0.6143	-0.4892	0.6376	0.3386	-0.0021	0.7456	0.2818	-0.4892	-0.0526	-0.0081	0.9091	0.0672	0.4261	0.5105	0.9666	-0.4250
'VmHWM'	0.9855	0.9799	1.0000	0.9676	0.9773	0.5955	-0.3872	0.6102	0.2227	-0.0025	0.6771	0.1693	-0.3872	-0.0548	-0.0121	0.9160	0.0775	0.3148	0.5149	0.9441	-0.4366
'VmRSS'	0.9389	0.9628	0.9676	1.0000	0.9416	0.5666	-0.3733	0.5484	0.2250	-0.0039	0.6400	0.1721	-0.3733	-0.0439	-0.0090	0.9488	0.0705	0.2986	0.5180	0.9113	-0.3905
'VmPTE'	0.9900	0.9906	0.9773	0.9416	1.0000	0.6324	-0.4569	0.6320	0.2254	0.0179	0.5241	0.1696	-0.4569	-0.0436	-0.0091	0.8910	0.0733	0.3525	0.5155	0.9640	-0.4437
'Threads'	0.6312	0.6102	0.5955	0.5666	0.6324	1.0000	-0.2254	0.4141	0.0125	0.4577	0.0512	0.2254	-0.0254	0.0113	0.3142	0.1696	0.2718	0.3848	0.6873	-0.3936	
'MemFree'	-0.4668	-0.4892	-0.3872	-0.3733	-0.4145	-0.2234	1.0000	-0.7343	-0.9881	-0.0007	-0.9064	-0.9645	1.0000	0.0015	0.0053	-0.0348	-0.0173	-0.9848	-0.1066	-0.4897	0.1354
'Buffers'	0.6530	0.6376	0.6102	0.5484	0.6179	0.4707	-0.7243	1.0000	0.5320	-0.0016	0.8927	0.5554	-0.7243	-0.1280	-0.0339	0.4969	0.1065	0.6364	0.2221	0.6362	-0.2983
'Cached'	0.3069	0.3386	0.2227	0.2250	0.2541	0.0847	-0.9681	0.5320	1.0000	0.0046	0.7825	0.9920	-0.9681	-0.0460	-0.0019	0.1906	0.0380	0.9797	0.0171	0.3376	-0.0470
'Active'	-0.0012	-0.0021	-0.0025	-0.0039	-0.0014	0.0128	-0.0007	-0.0116	0.0046	1.0000	-0.0005	0.0042	-0.0007	-0.0019	-0.0005	-0.0040	-0.0063	0.0005	-0.0032	-0.0004	0.0105
'Inactive'	0.7396	0.7454	0.6771	0.6400	0.6995	0.4577	-0.9064	0.8927	0.7825	-0.0056	1.0000	0.7631	-0.9064	-0.0285	0.0244	0.6131	0.1456	0.8547	0.3069	0.7533	-0.2894
'Dirty'	0.2508	0.2818	0.1693	0.1721	0.1958	0.0569	-0.9645	0.5354	0.9920	0.0046	0.7631	1.0000	-0.9645	-0.0823	0.0243	0.1280	-0.0696	0.9704	0.0244	0.2756	-0.0248
'WriteBack'	0.2254	0.2254	0.2254	0.2254	0.2254	0.2254	-0.2254	0.2254	0.2254	0.2254	0.2254	0.2254	-0.2254	0.2254	0.2254	0.2254	0.2254	0.2254	0.2254	0.2254	0.2254
'Mapped'	-0.0605	-0.0526	-0.0548	-0.0439	-0.0491	-0.0234	0.0615	-0.1290	-0.0468	-0.0019	-0.0285	-0.0281	0.0615	1.0000	0.4209	0.1885	0.7718	-0.0252	0.4869	0.0789	0.3806
'Slab'	-0.0113	-0.0081	-0.0121	-0.0090	-0.0087	-0.0013	0.0053	-0.0339	-0.0019	-0.0005	0.0244	-0.0243	0.0053	0.4209	1.0000	0.1166	0.4407	-0.0045	0.1859	0.0627	0.3344
'PageTables'	0.8858	0.9091	0.9160	0.9488	0.8910	0.5442	-0.3348	0.4969	0.1908	-0.0040	0.6131	0.1280	-0.3348	0.1885	0.1166	1.0000	0.3705	0.2739	0.6597	0.9179	-0.3488
'CommittedAS'	0.0696	0.0674	0.0775	0.0705	0.0753	0.1034	-0.0173	0.1065	-0.0380	-0.0063	0.1456	-0.0696	-0.0173	0.7718	0.4407	0.3705	1.0000	0.0287	0.5679	0.2463	0.0207
'NumOfAllocFH'	0.4055	0.4261	0.3143	0.2981	0.3525	0.2188	-0.9848	0.6364	0.9797	0.0059	0.8547	0.9704	-0.9848	-0.0252	-0.0045	0.2739	0.0287	1.0000	0.1008	0.4413	-0.1120
'proc-fd'	0.5016	0.5105	0.5149	0.5180	0.5155	0.3489	-0.1066	0.2221	0.0171	-0.0032	0.3069	-0.0344	-0.1066	0.4869	0.1859	0.6597	0.0379	0.1008	1.0000	0.5915	-0.1956
'avgThroughput'	0.9550	0.9666	0.9441	0.9113	0.9540	0.6373	-0.4897	0.6862	0.3376	-0.0004	0.7533	0.2756	-0.4897	0.0789	0.0627	0.5179	0.2463	0.4413	0.03515	1.0000	-0.0410
'avgElapsed'	0.4981	-0.4250	-0.4366	-0.3905	-0.4437	-0.3369	0.1354	-0.2983	-0.0470	0.0105	-0.2894	-0.0248	0.1354	0.0808	0.0344	-0.3468	0.0207	-0.1120	-0.1956	-0.4100	1.0000

Figura 2.2: Matrice di correlazione

E' possibile notare che le componenti WriteBack e MemFree sono perfettamente correlate. Si va quindi ad escludere la componente MemFree conservando WriteBack. Si conservano un totale di 20 componenti(colonne) e 4995 osservazioni(righe), 99.900 valori, con cui si proseguirà l'analisi.

Standardizzazione

Per fare si che l'analisi alle componenti principali (PCA) non risulti influenzata dal valore assoluto dai dati, è necessario eseguire una normalizzazione dei dati stessi. Si utilizza la normalizzazione z-score, in modo che ogni componente(colonna) abbia

media nulla e varianza unitaria.

$$z = \frac{x - \bar{x}}{\sigma_x} \quad (2.1)$$

2.2.2 Principal Component Analysis (PCA)

La prima tecnica che si applica ai dati così rielaborati è la Principal Component Analysis (PCA), permette di trovare le componenti principali di un workload usate successivamente per effettuare un cambio di base. Le componenti principali di un set di dati sono dei vettori direzione ortogonali che individuano una base ortonormale, nella quale le dimensioni(componenti) dei dati sono linearmente indipendenti. Inoltre, le componenti principali sono ordinate per quantità di varianza spiegata dalla componente stessa: la prima componente principale è la prima direzione che massimizza la varianza spiegata, la seconda componente è la seconda direzione che massimizza la varianza, ecc. La PCA è una procedura preliminare che serve ad effettuare quello che viene detto dimensionality reduction: si proietta ogni punto del workload in uno spazio costituito solo da alcune delle prime componenti principali, in modo da ottenere un dataset caratterizzato da una minore dimensionalità, conservando quanta più varianza possibile. Tramite JMP, la PCA è possibile eseguirla andando a fare un analisi multivariata alle componenti principali, ottenendo quindi tali autovettori e autovalori.

CAPITOLO 2. WORKLOAD CHARACTERIZATION

	1 principale	2 principale	3 principale	4 principale	5 principale	6 principale	7 principale	8 principale	9 principale	10 principale	11 principale	12 principale	13 principale	14 principale	15 principale	16 principale	17 principale	18 principale	19 principale	20 principale
'VmPeak'	0.30434	0.11963	-0.10770	0.00276	0.04478	0.07565	-0.04179	-0.00169	0.05963	0.12527	0.29311	0.39099	0.04309	0.01196	0.01804	-0.69891	0.27903	0.21278	-0.03775	0.0026
'VmSize'	0.30688	0.11007	-0.0769	0.00315	0.08893	0.08957	-0.07978	0.04226	0.06615	0.06209	0.09268	-0.12255	-0.29942	0.21468	-0.05025	0.26237	0.02551	-0.47944	0.00201	-0.00662
'VmHWM'	0.29438	0.16211	-0.11700	0.00149	0.08775	0.09349	-0.08505	-0.06367	0.06836	0.07785	0.00059	0.62533	0.30999	-0.0456	-0.12451	0.46432	-0.21186	-0.11756	0.00276	-0.00034
'VmRSS'	0.28659	0.16437	-0.10736	0.00230	0.15790	0.11642	-0.13559	0.01492	0.08845	-0.08833	-0.54776	-0.14215	0.00054	-0.0724	0.13653	0.16576	0.23618	0.61510	0.00191	0.00276
'VmLCK'	0.28652	0.17777	0.01529	0.00230	0.11125	0.09329	-0.08164	0.00468	0.08845	0.08836	-0.54776	-0.14215	0.00054	-0.0724	0.13653	0.16576	0.23618	0.61510	0.00191	0.00276
'Threads'	0.20935	-0.13632	-0.08474	0.00230	-0.27404	0.10721	0.73215	0.52981	0.04466	0.04466	0.07796	0.10454	0.00054	-0.0724	0.13653	0.16576	0.23618	0.61510	0.00191	0.00276
'Buffers'	0.24944	-0.14492	0.01621	-0.02039	-0.15767	0.08697	0.37331	-0.64424	-0.15614	0.31650	-0.19478	-0.12292	0.14713	0.24234	0.06212	-0.05204	0.02011	-0.02221	0.09827	0.24046
'Cached'	0.17554	-0.41314	0.11498	0.00458	0.03993	0.09178	-0.14500	0.21909	0.02833	-0.09504	0.07659	0.03851	-0.08484	-0.09142	-0.31721	0.00355	-0.01334	0.05264	0.29209	0.70548
'Active'	0.00771	0.03627	-0.02000	0.99864	-0.03921	-0.01195	-0.02261	-0.03339	-0.00267	0.00361	-0.03277	-0.01111	-0.01723	-0.00150	-0.00026	0.00004	-0.00013	0.00001	-0.00001	
'Inactive'	0.28863	-0.20992	0.06126	-0.00811	-0.04778	0.01704	0.12117	-0.21066	-0.04686	0.05658	0.02184	0.03722	-0.35921	-0.70356	-0.09736	0.03235	-0.05472	-0.05774	0.18967	-0.36169
'Dirty'	0.16081	-0.43625	0.10131	0.00250	0.02486	0.00920	-0.03080	0.13300	-0.01259	-0.04568	-0.12014	-0.03491	0.18772	0.43782	-0.03054	-0.06611	0.00324	0.06165	0.29532	-0.55876
'Writeback'	-0.22322	0.36612	-0.09317	0.00120	0.00468	0.01665	0.01919	-0.01343	0.01348	0.01105	0.05376	-0.00049	0.02037	0.00751	0.15673	0.01401	0.02733	-0.01258	0.88162	0.00380
'Mapped'	0.00335	0.14293	0.55805	0.00148	0.01025	-0.21614	-0.08164	0.14441	0.47426	0.59024	-0.17264	-0.03704	0.02621	-0.02299	0.00968	-0.02057	-0.01850	-0.02737	0.00001	-0.00012
'Spare'	0.27863	0.00008	-0.39668	-0.00008	-0.35876	0.81135	-0.19424	0.00000	-0.19424	-0.19424	-0.19424	-0.19424	-0.19424	-0.19424	-0.19424	-0.19424	-0.19424	-0.19424	-0.19424	
'PageTables'	0.27863	0.26822	0.00008	-0.35876	0.81135	-0.19424	-0.19424	-0.19424	-0.19424	-0.19424	-0.19424	-0.19424	-0.19424	-0.19424	-0.19424	-0.19424	-0.19424	-0.19424	-0.19424	
'Committed_AS'	0.05381	0.16103	0.54858	-0.00658	-0.02359	0.14659	0.31156	0.14488	-0.57038	0.02834	0.15428	0.08265	0.14409	0.05815	0.12129	0.13929	0.17807	0.00201	0.00100	
'NumOfAllocFH'	0.020561	-0.37914	0.111929	0.00485	-0.01738	-0.05675	-0.02246	0.17846	-0.07299	-0.06274	0.12440	0.09774	0.06689	0.02799	0.84344	0.10290	-0.00801	-0.05139	0.06586	-0.00012
'proc_fd'	0.16690	0.24060	0.02906	0.00326	0.16322	-0.30943	-0.15756	0.17711	-0.80191	0.12639	-0.03443	0.01448	-0.01132	-0.01457	-0.02030	-0.01010	-0.00221	0.00433	0.00336	0.00003
'avgThroughput'	0.30609	0.12389	0.00541	0.00510	0.05607	0.02575	-0.01088	0.03174	0.07127	-0.07038	0.42134	-0.58132	0.51913	-0.22187	-0.06355	0.03566	-0.01953	0.03053	0.00125	-0.00011
'avgElapsed'	-0.13731	-0.10299	0.14865	0.03804	0.81417	0.30099	0.43341	0.00254	-0.00970	0.05890	0.04648	0.01632	-0.00468	-0.00208	0.00565	-0.00297	0.00071	0.00099	0.00002	-0.00001

Figura 2.3: Autovettori PCA

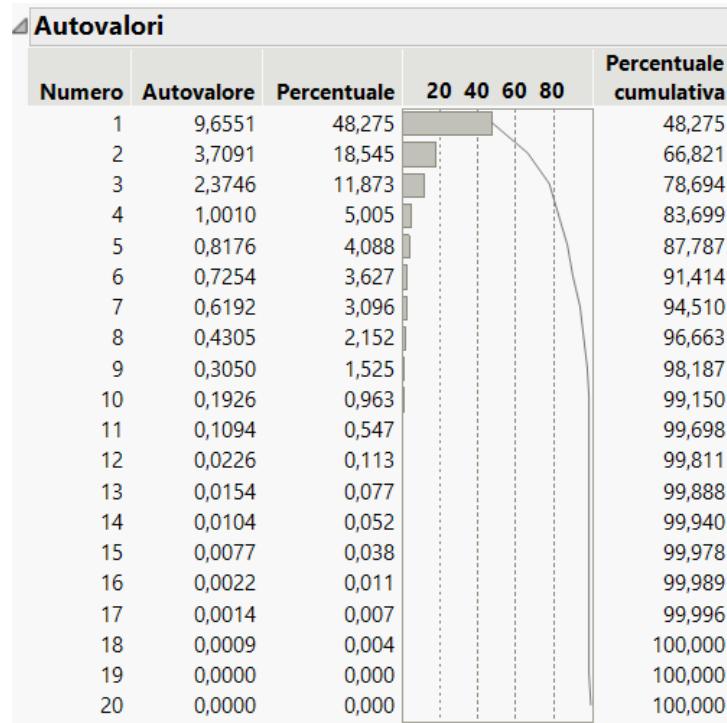


Figura 2.4: Autovalori PCA

Eseguita la PCA è possibile osservare la varianza spiegata da ogni componente principale come riportato in tabella. La colonna Percentuale rappresenta il rapporto tra varianza spiegata dalla i-esima componente e varianza totale del dataset, l'ultima colonna invece mostra la varianza percentuale cumulativa spiegata. In questo caso si è deciso di conservare le **prime 7 componenti con** una varianza

percentuale spiegata del 94.510% quindi accettabile. Vincoli sui costi influenzano il numero di componenti principali da conservare.

2.2.3 Clustering

Dopo aver ridotto la dimensionalità del workload (numero di colonne) si vuole ridurre anche la sua dimensione (numero di righe). A tal fine è possibile utilizzare la tecnica del clustering, la quale consiste nel raggruppare un insieme di elementi in modo che gli elementi nello stesso gruppo sono più simili tra loro rispetto agli elementi di altri gruppi. Tra le diverse tecniche di clustering si è scelto di utilizzare un approccio agglomerativo detto metodo di Ward, tale tecnica di clusterizzazione di per sé non introduce ulteriore perdita di varianza perché raggruppa le istanze minimizzando la possibile devianza nei cluster. Esso un metodo di clustering gerarchico che minimizza la devianza intra-cluster. Minimizzando tale quantità si minimizza anche la varianza intraccluster, che rappresenta proprio la porzione di varianza persa applicando il clustering rispetto alla varianza totale del dataset. L'obiettivo è quello di identificare classi che rappresentino comportamenti funzionali del sistema differenti, riducendo allo stesso tempo la dimensione dell'insieme dei dati. Lo scopo è quello di minimizzare la variabilità interna ad ogni cluster (devianza intraccluster) e massimizzare quella esterna (devianza intercluster). Secondo il metodo di Ward, all'inizio ogni elemento del dataset rappresenta un cluster diverso. Vengono poi costruiti nuovi cluster in maniera iterativa, unendo ad ogni iterazione i due cluster più vicini. Per la distanza tra



cluster si usa quella di Ward definita:

$$d(P, Q) = 2 \frac{|P| |Q|}{|P| + |Q|} \| \bar{x}_P - \bar{x}_Q \| \quad (2.2)$$

dove $|Q|$ è la cardinalità del cluster Q e \bar{x}_Q è il suo centroide definito come:

$$\bar{x}_Q = \frac{1}{|Q|} \sum_{i=1}^{|Q|} x_i \quad (2.3)$$

x_i è l'i-esimo elemento del cluster Q. Lo stesso vale per P. Grazie a JMP è possibile eseguire facilmente il clustering sulle componenti principali conservate, ottenendo in output il dendogramma. Eseguire il clustering sulle componenti principali PCA ha due vantaggi:

- Il clustering è applicato ad un workload avente un numero minore di componenti(colonne)
- Si prevengono errori nel clustering, essendo le componenti principali correlate

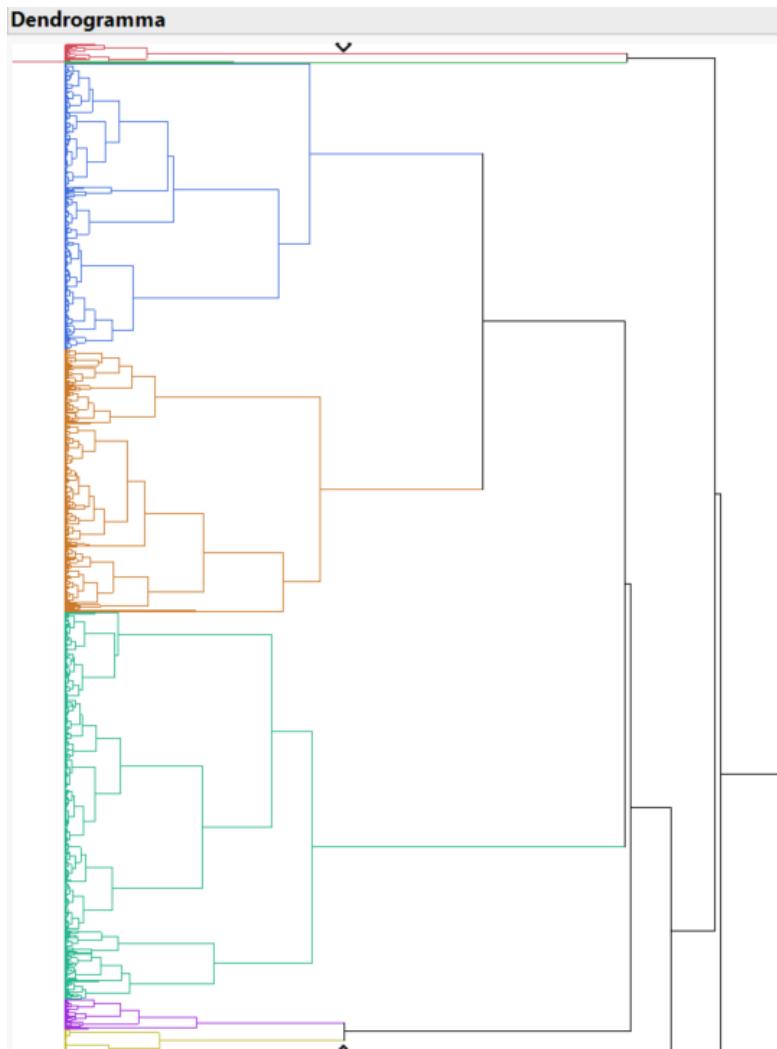


Figura 2.5: Dendrogramma

Si prosegue analizzando la cronologia di clusterizzazione al fine di decidere il numero ottimale di cluster. Si decide di utilizzare 9 cluster, anche se la scelta andrebbe fatta valutando la cronologia di clusterizzazione, nel caso di 7 cluster si avrebbe un aumento di distanza maggiore nel caso di aggiunta di un cluster, e una diminuzione sostanziale di distanza con la rimozione di uno di essi, si decide comunque di utilizzare 9 cluster per conservare una maggiore devianza. In definitiva, si decide di utilizzare 9 cluster.

CAPITOLO 2. WORKLOAD CHARACTERIZATION

tiva si avranno delle etichette per ogni istanza del workload reale che definiranno il cluster di appartenenza.

distanza = devianza intra-cluster

Cronologia di clusterizzazione			
Numero di cluster	Distanza	Leader	Subordinato
1	70,19373692	1	90
2	69,52862696	1	92
3	64,85206556	92	4943
4	60,57731541	92	2737
5	60,12229624	1	84
6	59,95621022	92	96
7	44,75813132	92	1873
8	29,84367272	2737	2885
9	27,21749845	1873	1931
10	26,43222079	96	193
11	26,17793531	92	109
12	23,34276535	1931	3738
13	22,84437429	109	719
14	22,08914670	96	102
15	18,04891125	84	91
16	15,90771326	193	1863
17	14,76285279	1931	1932
18	14,68407166	102	739
19	13,99886127	2737	2844
20	13,92700225	3738	4140

Figura 2.6: Cronologia clustering

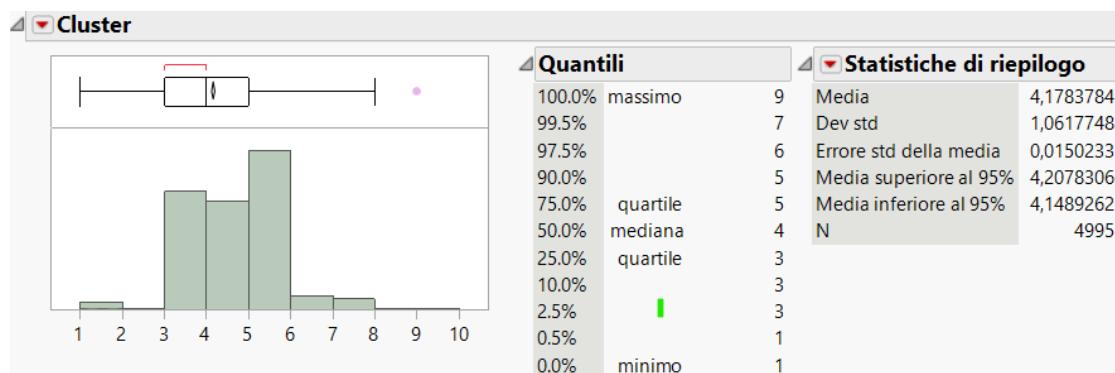


Figura 2.7: Distribuzione istanze cluster

Da tale analisi si vede come la distribuzione delle istanze sia sbilanciata, dal momento che alcuni cluster sono decisamente più ricchi di altri; ciò porta a dire che

sarà cruciale la scelta delle istanze per la caratterizzazione del workload sintetico per alcuni cluster.

2.2.4 Calcolo delle devianze

L'obiettivo della caratterizzazione è quello di definire un workload sintetico che sia rappresentativo di quello reale, in termini di variabilità dei dati. Per questo motivo, è fondamentale osservare come entrambe le tecniche usate nel processo di caratterizzazione vanno ad impattare sulla varianza del workload. Più precisamente, piuttosto che considerare la varianza si preferisce studiare la devianza, per motivi legati alla scelta della tecnica di clustering. Si considerano le prime 7 componenti della PCA e 9 cluster.

Devianza totale workload reale

$$D_{tot} = \sum_{i=1}^N \|x_i - \bar{x}\|^2 = 99899.99999 \quad (2.4)$$

dove N è il numero di istanze(righe) del workload reale, x_i è l'i-esima istanza, \bar{x} è la media delle istanze.

Devianza post PCA

Avendo conservato solo le prime 7 delle componenti principali la devianza del workload rappresentato in questo nuovo spazio sarà sicuramente cambiata. x_{PCAi} rappresenta l'i-esima istanza del workload nel nuovo spazio di stato, \bar{x}_{PCA} invece

è la media delle istanze.

$$D_{postPCA} = \sum_{i=1}^N \|x_{PCAi} - \bar{x}_{PCA}\|^2 = 94396.81999606364 \quad (2.5)$$

E' possibile verificare che il rapporto tra devianza totale del workload pre PCA e devianza totale dopo aver conservato le prime 7 componenti della PCA, è pari alla varianza che si è scelto di preservare:

$$\frac{D_{postPCA}}{D_{tot}} = 0.9449131 \simeq 94.5\% \quad (2.6)$$

Devianza post Clustering

L'operazione di clustering modifica la devianza dei dati. In particolare abbiamo due contributi, la devianza *intra-cluster* e quella *inter-cluster*, rappresentano rispettivamente la devianza tra i dati di uno stesso cluster e tra i dati di due cluster differenti.

$$D_{PCA} = D_{intra} + D_{inter} \quad (2.7)$$

Dal momento che per creare il workload sintetico si considera un campione casuale per ogni cluster, D_{intra} rappresenta la **devianza persa** a causa del clustering, mentre D_{inter} rappresenta la **devianza preservata**

Si calcolano i due contributi come segue:

$$D_{intra} = \sum_{k=1}^K \sum_{i=1}^{n_k} \|x_i - \bar{x}_k\|^2 = 12058.598990 \quad (2.8)$$

$$D_{inter} = \sum_{k=1}^K n_k \|\bar{x}_k - \bar{x}\|^2 = 82338.221005 \quad (2.9)$$

dove i termini indicano

- K è il numero di cluster
- n_k è il numero di elementi del cluster k
- \bar{x}_k è il centroide del cluster k
- \bar{x} è la media degli elementi del dataset nello spazio trasformato ridotto
- x_i è l'i-esimo elemento del cluster k

La somma dei due contributi sarà pari alla devianza dopo l'analisi PCA

$$D_{PCA} = D_{intra} + D_{inter} \rightarrow 94396.8199 = 12058.598990 + 82338.221005 \quad (2.10)$$

Perdita di devianza

E' possibile calcolare la devianza totale persa a seguito dell'analisi PCA e clustering, seguendo due approcci diversi:

1. La porzione di devianza persa può essere vista come la somma tra la devianza persa a causa della PCA e quella persa a causa del clustering(intra-cluster) relativa alla devianza preservata dalla PCA

$$\%D_{loss} = (1 - \%D_{postPCA}) + \%D_{intra} * \%D_{postaPCA} = 0.1757 \simeq 17.6\% \quad (2.11)$$

2. La porzione di devianza persa può essere vista come la devianza totale meno la devianza preservata, che a sua volta è data dal prodotto tra la devianza preservata dalla PCA e quella preservata dal clustering (devianza inter-cluster)

$$\%D_{loss} = 1 - \%D_{inter} * \%D_{postaPCA} = 0.1757 \simeq 17.6\% \quad (2.12)$$

In conclusione, eseguendo la workload characterization considerando 7 componenti principali e fissando il numero di cluster a 9, si perde circa il 17.6% della varianza del workload reale.

2.3 Analisi delle devianze

Grazie al clustering è possibile ridurre la dimensione del workload, ma la perdita di devianza dipende dal numero di cluster ma anche dal numero di componenti PCA che si vanno ad utilizzare. E' necessario trovare il numero di componenti principali e clusters che assicura il miglior trade-off tra devianza persa e riduzione della dimensione del workload

2.3.1 Devianza persa vs Numero di componenti principali

Per valutare la devianza persa e la variazione di quest'ultima al variare del numero di componenti principali ed al numero di cluster basta ricalcolare le devianze al variare dei due parametri. Si è deciso di far variare il numero di componenti principali usate nell'intervallo [3,7] ed il numero di cluster nell'intervallo [1,20]. A questo punto è possibile osservare come varia la devianza persa in funzione del

numero di cluster, per diversi valori di componenti principali come mostrato in figura

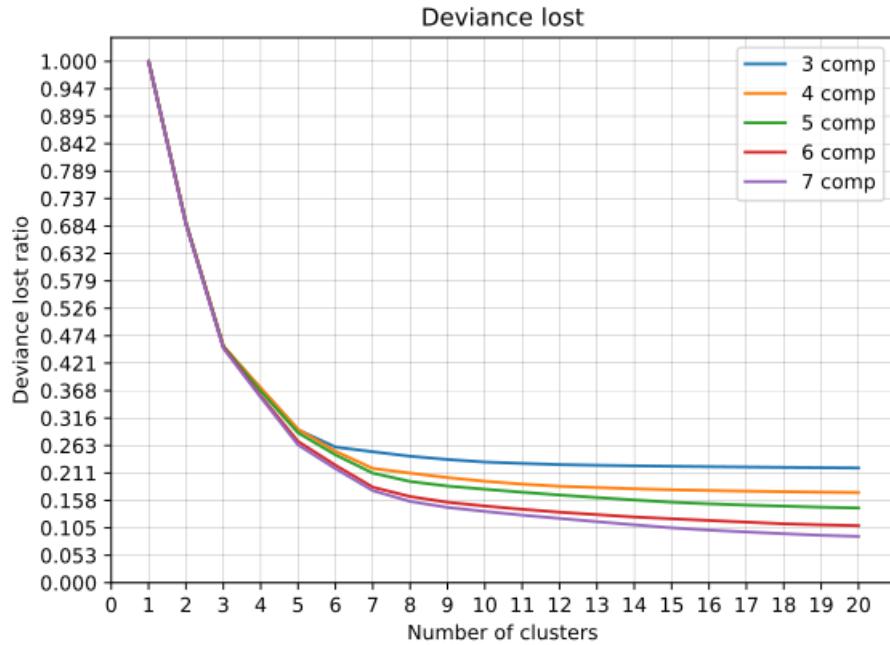


Figura 2.8: Varianza persa

La perdita di devianza presenta un andamento monotono decrescente all'aumentare del numero di cluster, indipendentemente dal numero di componenti principali. Tale risultato è ragionevole dal momento che aumentando il numero di clusters si vanno a considerare più osservazioni e di conseguenza si preserva una maggiore quantità di devianza. Un valore ottimale di cluster è quindi in corrispondenza del gomito di tali curve, e come è possibile notare, il gomito si ha come visto intorno al cluster 7. Per un numero di clusters strettamente minore di 7, la quantità di devianza persa non dipende molto dal numero di componenti principali considerate. Al contrario, se il numero di cluster è maggiore o uguale a 7, la perdi-

ta di devianza diventa minore all'aumentare del numero di componenti principali, visto l'andamento lineare che si ottiene dal grafico.

2.3.2 Devianza persa vs Dimensione del workload sintetico

Un'ulteriore osservazione che è possibile fare è riguardo la dimensione del WL sintetico che deriva dall'analisi. Per dimensione di workload sintetico, si intende il rapporto tra numero di valori del workload sintetico e numero di valori del workload reale. Idealmente, si vorrebbe che sia la devianza persa che la dimensione del workload sintetico siano quanto più bassa possibili. Tuttavia, la devianza persa decresce con l'aumentare del numero di clusters (e di componenti principali) mentre la dimensione del workload sintetico aumenta con l'aumentare degli stessi parametri. Per questo motivo, è possibile considerare il prodotto tra devianza persa e dimensione del workload sintetico ed osservare quali sono i valori di cluster e componenti principali che minimizzano tale quantità. Graficando i vari contributi si è ottenuto in grafico seguente.

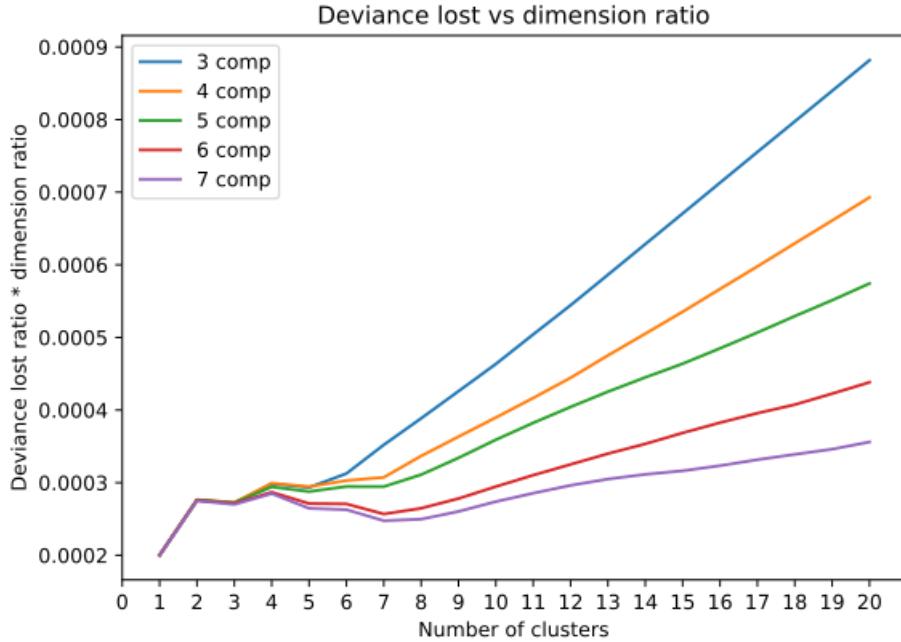


Figura 2.9: Varianza persa in funzione del numero di clusters e componenti principali

Tale risultato sta a significare che una scelta elevata di cluster porterebbe ad un dimensionamento elevato del workload sintetico, con una conseguente scarsa riduzione di dimensionalità, che è l'obiettivo di tale analisi. La scelta fatta di considerare un numero di cluster pari a 9 risulta valida, in quanto si ottiene un buon trade-off tra la dimensionalità del workload e la perdita di devianza.

2.3.3 Workload sintetico

Partendo da un workload di 4995 osservazioni con 24 componenti (119.880 valori), è stato ottenuto un workload sintetico costituito da sole 9 osservazioni (216

CAPITOLO 2. WORKLOAD CHARACTERIZATION

valori), ognuna delle quali è un campione estratto dal relativo cluster. Tale caratterizzazione avviene o scegliendo da ogni cluster un elemento randomico per andare a caratterizzarne le istanze, o, come avverrà in questa analisi, andare ad identificare il centroide reale del cluster, cioè l'istanza che di trova più vicino al centroide virtuale calcolato durante la fase di definizione delle devianze. Nello spazio trasformato dalle operazioni di PCA, l'analisi dei 9 cluster scelti ha portato all'individuazione di 9 centroidi virtuali, che sono riportati in tabella. Si è quindi proceduto a calcolare la distanza euclidea di tutte le istanze dai centroidi dei cluster, per poi identificare l'istanza che risultasse più vicina.

Principale1	Principale2	Principale3	Principale4	Principale5	Principale6	Principale7	cluster
-17.712506	8.276706	-6.073880	-0.151787	-3.609697	0.910962	-0.033210	1
-6.280465	11.580185	17.847463	0.079177	2.964769	-8.512313	-0.337308	2
-1.831781	4.731509	13.189867	0.053565	2.008693	-5.889389	-2.403400	3
0.299274	-0.463701	-0.189017	-0.109679	-2.000203	-0.746652	0.021152	4
-1.879731	1.359506	4.113281	0.047870	1.177306	-1.574176	-0.769960	5
5.800611	2.114084	-0.540152	0.088480	1.524919	0.662554	1.552007	6
7.575456	3.060815	-1.095833	-0.088896	0.761415	0.579470	-2.683569	7
-2.123175	-0.797758	-0.043829	64.822920	-1.192900	-0.449301	-1.710205	8
6.353504	20.541277	66.895610	0.061951	-20.732429	41.702500	-0.592811	9

Figura 2.10: Workload sintetico PCA

CAPITOLO 2. WORKLOAD CHARACTERIZATION

cluster	1	2	3	4	5	6	7	8	9
'VmPeak'	144012	153484	170344	177444	170344	201020	215352	172388	170344
'VmSize'	126424	151436	170340	173608	170340	199992	206136	170340	170340
'VmHWM'	14224	26612	31148	35624	31920	49084	61784	32696	31144
'VmRSS'	14224	26468	31140	33952	31912	48704	53204	31460	31140
'VmPTE'	108	160	200	208	200	292	312	200	200
'Threads'	9	45	26	82	51	161	26	33	52
'MemFree'	5621452	5160008	4720436	4569760	4673128	4541144	4538104	4635532	4655664
'Buffers'	29064	83244	97808	191448	98276	203636	204356	132072	89480
'Cached'	334600	656460	1064968	1117920	1110480	1118352	1118452	1117192	1090572
'Active'	0	0	0	0	0	0	0	111	0
'Inactive'	131152	230156	366496	469124	406568	513508	520160	412036	541220
'Dirty'	300072	597448	880988	926516	888336	909524	908192	921028	773332
'Writeback'	5621452	5160008	4720436	4569760	4673128	4541144	4538104	4635532	4655664
'AnonPages'	8159224	8159224	8159224	8159224	8159224	8159224	8159224	8159224	8159224
'Mapped'	156	221152	192588	140	57796	152	156	164	324400
'Slab'	0	0	0	0	0	0	0	0	1760
'PageTables'	67408	87804	84604	86292	86164	101032	105544	83800	134500
'Committed_AS'	22840	25312	24312	23844	23948	23848	23848	23740	29956
'NumOfAllocFH'	26160	89796	109564	112720	109840	112868	111100	109580	106788
'proc-fd'	6968	7796	7576	7212	7320	7300	7320	7208	8028
'avgThroughput'	268820	304888	315392	319440	319164	340920	349636	316560	365264
'avgElapsed'	510	2040	1530	510	1530	1530	510	1530	2040
'avgLatency'	0	0	0	0	0	0	0	0	0
'Errors'	2	2	2	2	2	2	2	2	2

Figura 2.11: Workload sintetico

Capitolo 3

Web Server

3.1 Capacity Test

3.1.1 Traccia

Effettuare l'analisi prestazionale di un Web Server attraverso le seguenti fasi:

- Analizzare il throughput e il response time all'incremento dei valori di load applicati al web server
- Effettuare per ogni valore di load 3 o più ripetizioni
- Memorizzare i dati ottenuti dalle osservazioni per ogni valore di load calcolandone la media o la mediana per il throughput e il response time
- Graficare l'andamento del throughput e del response time all'incrementare dei valori di load
- Individuare il punto di knee e di usable capacity

3.1.2 Introduzione

Il primo step per effettuare l'analisi di un web server è definire il sistema da analizzare.

Il **System Under Test (SUT)** è un server Linux eseguito su una macchina virtuale Oracle VM Virtual BOX. Per analizzare tale web server, le richieste sono state effettuate tramite il tool Apache JMter, eseguito su un client Windows avente le caratteristiche riportate in Tabella 3.1.

Client	
CPU	i7-8750H
Core	6
RAM	16GB
SO	Windows 10 Home 64-bit

Tabella 3.1: Specifiche client

Utilizzando un singolo client per effettuare le richieste il System Under Test è stato dotato di specifiche minime, riportate nella Tabella 3.2.

Server	
CPU	i7-8750H
Core	1
RAM	1GB
SO	Ubuntu 20.4.1 LTS 64-bit

Tabella 3.2: Specifiche server

In particolare l'analisi si focalizza sullo studio di un componente specifico del

sistema, ovvero un server Apache 2.4 che rappresenta il **Component Under Study** (CUS).

Il web server espone una serie di risorse ad un client accessibili tramite richieste http. Per ricreare uno scenario reale si è scelto di utilizzare tre tipi di risorse, classificate sulla base della loro dimensione. Per ogni tipologia sono presenti tre file distinti. In Tabella 3.4 sono riportate le caratteristiche di ogni singola risorsa.

Tipo	Server	Dimensione
Small	small1.png	614kb
	small2.png	614kb
	small3.png	512kb
Medium	medium1.png	4.2MB
	medium2.png	3.1MB
	medium3.png	2.1MB
Large	large1.png	9.4MB
	large2.png	7.3MB
	large3.png	7.3MB

Tabella 3.3: Specifiche risorse

3.1.3 Esecuzione del test

Per l'esecuzione del *capacity test* il client Apache JMeter è stato configurato nel seguente modo:

- In tutti i casi le richieste sono state effettuate da **50 threads** appartenenti ad **un unico Thread Group**
- Ciascun thread ad ogni attivazione sceglie casualmente mediante un Random Controller una tipologia di risorsa

- Il numero di richieste effettuate dal client viene imposto attraverso un Constant Throughput Timer CTT richieste al minuto
- La durata di ogni test è stata settata a 5 minuti

I test sono stati eseguiti utilizzando i seguenti valori di CTT: 0, 100, 300, 600, 1000. Per ogni valore fissato di CTT sono state effettuate 3 ripetizioni, di ogni ripetizione sono stati calcolati throughput e tempo di risposta medio. Successivamente, per ogni test sono state calcolate le medie di throughput e tempo di risposta su tutte le ripetizioni.

A partire da questi valori, sono stati tracciati tre grafici in cui è riportato l'andamento di throughput, response time e la potenza al variare del carico. Attraverso i grafici ottenuti è possibile determinare due valori significativi di capacità del server, noti come:

- **Knee capacity:** il punto dopo il quale il response time cresce rapidamente al crescere del carico ed il guadagno del throughput è piccolo.
- **Usable capacity:** il punto in cui si ottiene il massimo throughput senza superare una determinata soglia del response time. Tipicamente tale punto si trova in corrispondenza del punto di saturazione del throughput

Il grafico della potenza, definita come il rapporto tra il throughput e il response time, ci permette di individuare con maggiore facilità i due punti di interesse. Tipicamente la *knee capacity* si trova in corrispondenza dei punti di massimo della curva, mentre la *usable capacity* è situato vicino il punto di annullo della curva.

Di seguito in Figura 3.1 sono stati riportati i grafici ottenuti.

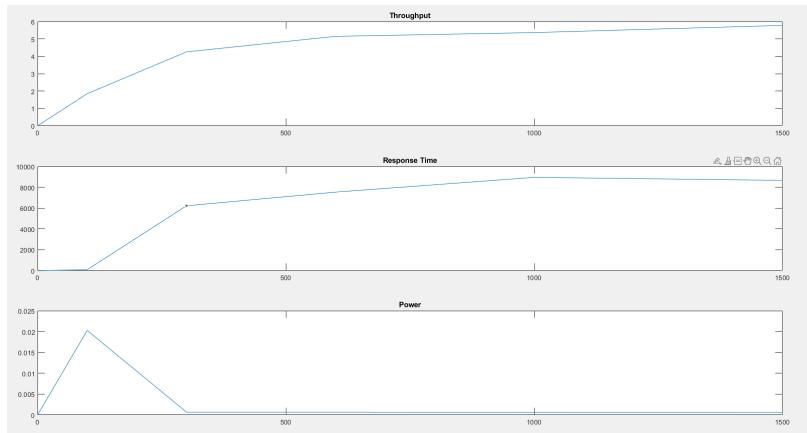


Figura 3.1: Risultati Capacity Test

Notiamo come la *knee capacity* può essere individuata in corrispondenza nel valore **CTT = 100**, mentre la *usable capacity* al punto **CTT = 1000**.

Fairness index

Possiamo anche procedere al calcolo della *Fairness*. La *Fairness* è una funzione compresa tra 0 ed 1, il cui valore decresce all'aumentare di disparità, e raggiunge il suo massimo, ovvero 1, in caso di massima equità, calcolabile come:

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

Dove gli x_i rappresentano il throughput normalizzato, ovvero, il rapporto tra il throughput misurato e quello nominativo. Nel nostro caso:

Measured Throughput : (100, 255, 321, 322)

Fair Throughput : (100, 300, 600, 1000)

L'indice di *Fairness* è pari quindi a **0.94**, possiamo quindi giungere alla conclusione che il sistema è sufficientemente paritario.

3.2 Workload characterization

La workload characterization è il processo di creazione di un workload sintetico, rappresentativo di un workload reale. In altre parole, si vuole estrarre un modello statistico del workload reale che ne rappresenti la variabilità. Sono state eseguite le seguenti fasi:

- **Fase 1**

- Creazione del workload reale:
 - * raccolta dei parametri di alto livello (HL)
 - * raccolta dei parametri di basso livello (LL)
- Caratterizzazione del workload reale:
 - * caratterizzazione dei parametri di alto livello (HL_c)
 - * caratterizzazione dei parametri di basso livello (LL_c)

- **Fase 2**

- Applicazione del workload sintetico di alto livello
 - * raccolta dei parametri di basso livello (LL')
- Caratterizzazione del workload sintetico:
 - * caratterizzazione dei parametri di basso livello (LL'_c)

- **Fase 3**

- Validazione del workload sintetico di alto livello:

- * Verificare che il workload sintetico (di basso livello) LL'_c sia rappresentativo del workload reale (di basso livello) LL_c con un test di ipotesi

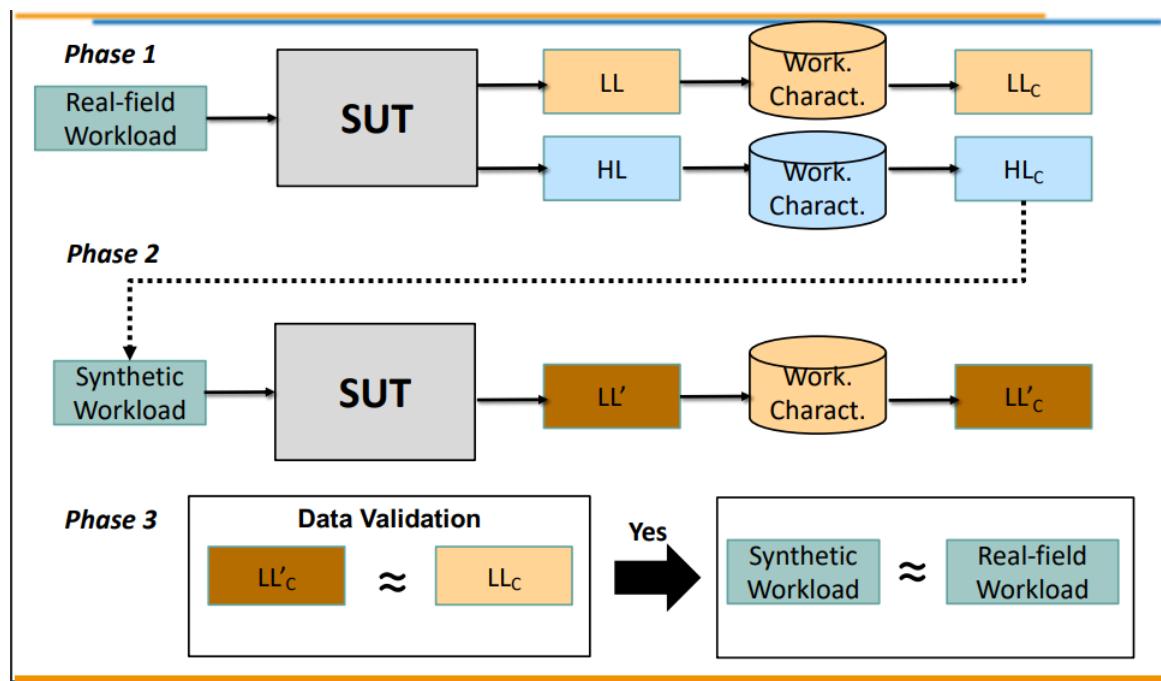


Figura 3.2: Operazioni Workload Characterization

3.2.1 Fase1

Generazione workload reale

Al fine di simulare l'imposizione di un carico reale al sistema si è cercato di diversificare quanto più possibile le richieste effettuate dal client. Si è deciso quindi di definire **quattro Thread Group**, ognuno dei quali effettua casualmente, attraverso un **Random Controller**, una delle tre tipologie di richieste. I **quattro Thread Group** si differenziano sulla base del numero di utenti e del numero di richieste al

minuto effettuate dal singolo utente, imposto tramite un **Constant Throughput Timer**.

La configurazione dei singoli Thread Group è riportata in tabella.

CTT				
Nome	Throughput	Utenti	Ramp-Up Period	Dimensione risorse
TG1	800	25	15	614 KB 4.2 MB 9.4 MB
TG2	100	40	15	614KB 3.1MB 7.3MB
TG3	300	50	25	512KB 2.1MB 7.3MB
TG4	600	60	30	512KB 4.2MB 8.4MB
TG5	500	50	25	512KB 2.1MB 8.4MB

Tabella 3.4: Configurazione Apache Jmeter

Una volta definita la configurazione, è stato eseguito un test della durata di cinque minuti, facendo lavorare i thread group in maniera sequenziale per un minuto ciascuno.

Raccolta dei parametri di alto livello (HL)

Tramite JMeter, all'atto dell'esecuzione del test si è scelto di raccogliere i parametri di alto livello sfruttando il **componente Simple Data Writer**, che crea un file di output contenente dei parametri specificati dall'utente. Si è scelto di usare tale componente rispetto ad altri che potevano fornire informazioni maggiori, perché è poco oneroso in termini di memoria lato client. È stato ottenuto un workload

reale con **1553 istanze e 14 componenti**. Sono stati scelti i seguenti parametri da salvare:

- **timeStamp**: istante temporale in cui la richiesta viene effettuata
- **elapsed**: : tempo trascorso tra l'invio della richiesta del client e la ricezione dell'ultimo pacchetto della risposta del server
- **responseCode**: modo in cui una richiesta HTTP è stata completata
- **responseMessage**: esito della richiesta
- **threadName**: nome del thread group a cui appartiene il thread che ha effettuato la richiesta
- **success**: esito della richiesta
- **bytes**: byte ricevuti dal client
- **sentbytes**: byte inviati dal client
- **grpThread**: numero di thread attivi all'interno del gruppo
- **allThread**: numero di thread attivi in tutti i thread group al momento della richiesta
- **URL**: URL della risorsa richiesta al server
- **latency**: : tempo trascorso tra l'invio della richiesta del client e la ricezione del primo pacchetto della risposta del server
- **idleTime**: numero di millisecondi di inattività
- **Connect**: tempo necessario per instaurare la connessione con il server

Raccolta dei parametri di basso livello (LL)

Durante l'esecuzione del test sono stati raccolti diversi parametri di basso livello attraverso il comando:

vmstat -n 1 305

Esso ha permesso di ricavare il workload di basso livello (LL), contenente informazioni relative allo stato del server. Nello specifico, per la durata di 305 secondi, con un periodo di campionamento di 1 secondo, sono stati raccolti 17 parametri, tra cui vale la pena menzionare:

- **Memory**

- *free*: memoria libera
- *buff*: memoria usata come buffer
- *cache*: memoria usata come cache

- **System**

- *in*: numero di interruzioni al secondo
- *cs*: numero di context-switch al secondo

- **CPU**

- *us*: percentuale di tempo speso in esecuzione di codice di livello utente
- *sy*: percentuale di tempo speso in esecuzione di codice di livello kernel
- *id*: percentuale di tempo di idle
- *id*: percentuale di tempo speso in attesa di operazioni di I/O

Caratterizzazione del workload reale

Per effettuare la caratterizzazione dei workload, sia di alto che di basso livello, sono state effettuate le seguenti fasi:

- **Preprocessing:** eliminazione di eventuali features nominali, costanti o perfettamente correlate
- **PCA**
- **Clustering**
- **Creazione del workload sintetico:** selezione dell'istanza da ognuno dei cluster

Caratterizzazione dei parametri di alto livello (HL_c)

Secondo le operazioni precedentemente descritte, è stata applicata la PCA al dataset costituito dai seguenti parametri di alto livello: *Elapsed, Byte, sentByte, allThreads, Latency, Connect.*

Successivamente, è stata effettuata una clusterizzazione gerarchica del dataset costituito dalle prime **5 componenti principali**, selezionando **30 cluster**.

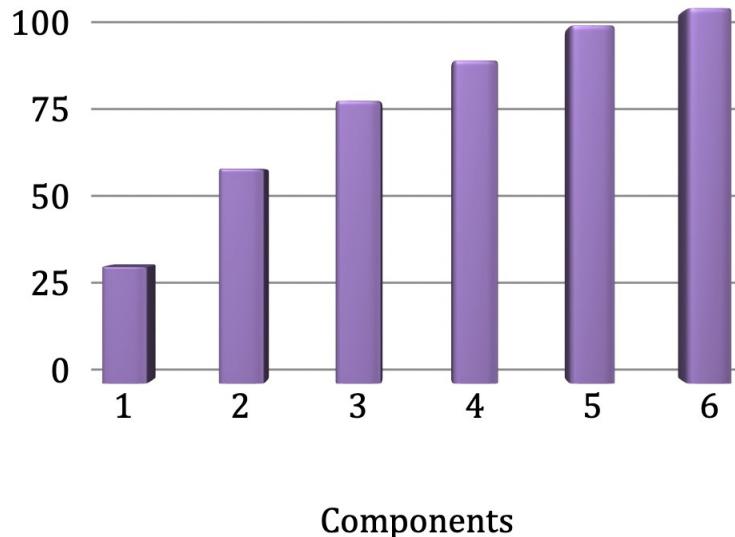


Figura 3.3: Devianza cumulativa spiegata

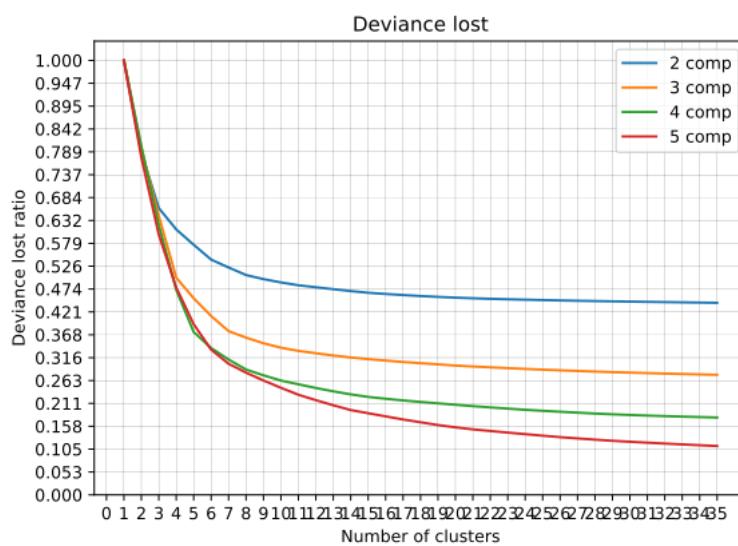


Figura 3.4: Analisi devianza

In Figura 3.4 è possibile osservare l'andamento della devianza persa in funzione del numero di cluster, al variare delle componenti principali. Con i parametri scelti

(5 componenti principali e 30 cluster) viene preservato circa l'88% di devianza.

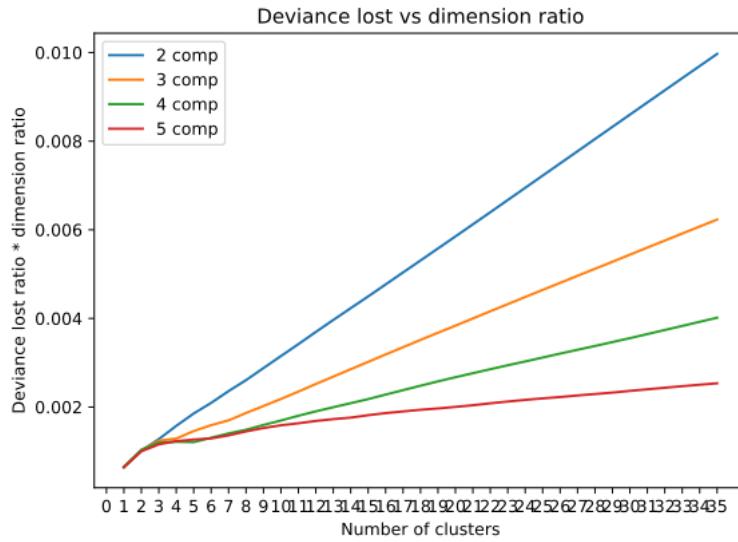


Figura 3.5: Analisi devianza

DEV_TOT	DEV_POST_PCA	DEV_POST_PCA_CL	DEV_LOST
9306	95%	87.8%	12.2%

Tabella 3.5: Analisi devianze

Come ultimo step è stato costruito il workload sintetico, estraendo un elemento da ognuno dei cluster. L'estrazione non è stata casuale, ma si è ragionato sulle caratteristiche dei cluster. In particolare, per ogni cluster è stata individuata la tipologia di risorsa più frequente. Successivamente è stata estratta la richiesta associata alla risorsa più frequente della data tipologia. Individuata la richiesta si è tenuto conto anche del Thread Group a cui appartiene il thread che l'ha generata. In conclusione, il workload sintetico di alto livello (HL_c) è costituito dalle richieste riportate in Tabella 3.6:

Cluster	URL
1	http://192.168.56.101/small4.jpg
2	http://192.168.56.101/small1.jpg
3	http://192.168.56.101/medium2.jpg
4	http://192.168.56.101/large3.jpg
5	http://192.168.56.101/large2.jpg
6	http://192.168.56.101/medium1.jpg
7	http://192.168.56.101/medium5.jpg
.	.
.	.
.	.
9	http://192.168.56.101/medium3.jpg
.	.
.	.
.	.
13	http://192.168.56.101/small5.jpg
.	.
.	.
.	.
22	http://192.168.56.101/large5.jpg
23	http://192.168.56.101/large1.jpg
.	.
.	.
.	.
25	http://192.168.56.101/large4.jpg
.	.
.	.
.	.
30	http://192.168.56.101/medium4.jpg

Tabella 3.6: Richieste costituenti il workload sintetico

Caratterizzazione dei parametri di basso livello (LL_c)

Secondo le operazioni precedentemente descritte, è stata applicata la PCA al dataset costituito dai seguenti parametri di basso livello: *procs_r, procs_b, memory_free, memory_buff, memory_cache, swap_si, swap_so, io_bi, io_bo, system_in, system_cs, cpu_us, cpu_sy, cpu_id, cpu_wa.*

Successivamente, è stata effettuata una clusterizzazione gerarchica del dataset costituito dalle prime **10 componenti principali, selezionando 20 cluster.**

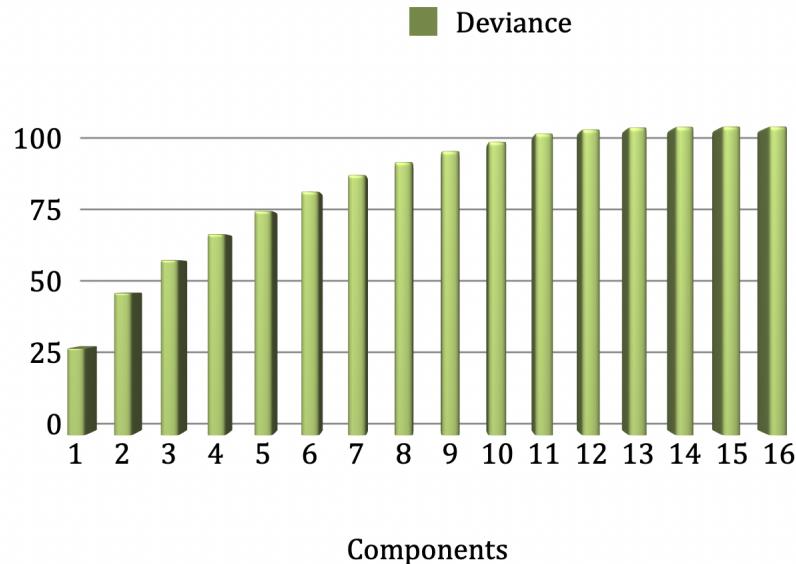


Figura 3.6: Devianza cumulativa spiegata

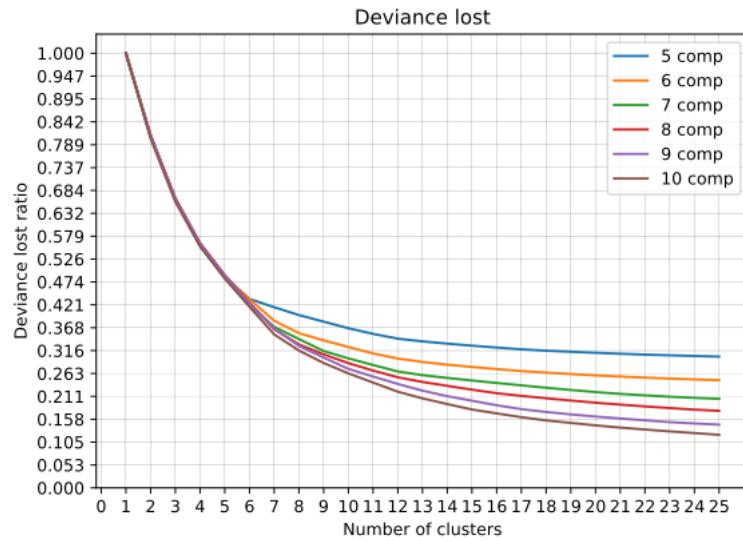


Figura 3.7: Analisi devianza

In Figura 3.7 è possibile osservare l'andamento della devianza persa in funzione del numero di cluster, al variare delle componenti principali. Con i parametri scelti (10 componenti principali e 20 cluster) viene preservato circa l'85% di devianza.

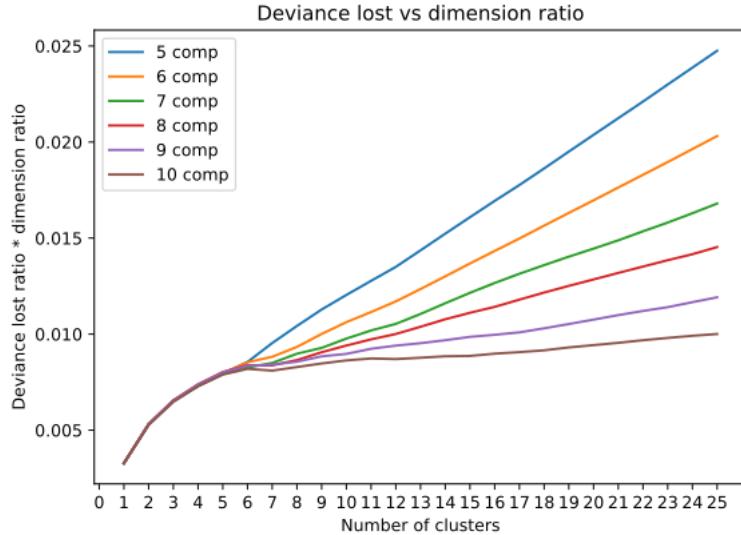


Figura 3.8: Analisi devianza

DEV_TOT	DEV_POST_PCA	DEV_POST_PCA_CL	DEV_LOST
4864	95%	85.1%	14.9%

Tabella 3.7: Analisi devianze

Infine, è stato costruito il workload sintetico, estraendo un elemento da ognuno dei cluster. In questo caso l'estrazione è stata del tutto casuale dato che, a differenza dell'alto livello, non potevamo contare il numero di occorrenze di un qualsiasi parametro.

3.2.2 Fase2

Applicazione del workload sintetico di alto livello

Dopo aver eseguito le operazioni di PCA e Clustering sul workload reale di alto livello, è necessario andare ad applicare il workload HL_c al web server, e raccogliere nuovamente i parametri di basso livello, per effettuare un confronto statistico e

validare la bontà del workload sintetico. Per farlo si è creato un nuovo Test Plan con le richieste HTTP contenute all'interno di HL_c da sottomettere al server.

Raccolta dei parametri di basso livello (LL'_c)

Durante l'esecuzione del test sono stati raccolti nuovamente i parametri di basso livello attraverso il comando: `vmstat -n 1 305`.

Caratterizzazione dei parametri di basso livello (LL'_c)

È stata effettuata la stessa workload characterization, dei parametri di basso livello risultanti dall'applicazione del workload reale, ai parametri di basso livello raccolti precedentemente e quindi risultanti dall'applicazione di quello sintetico. Quindi, è stato scelto lo stesso numero di componenti principali (10) e di cluster (20).

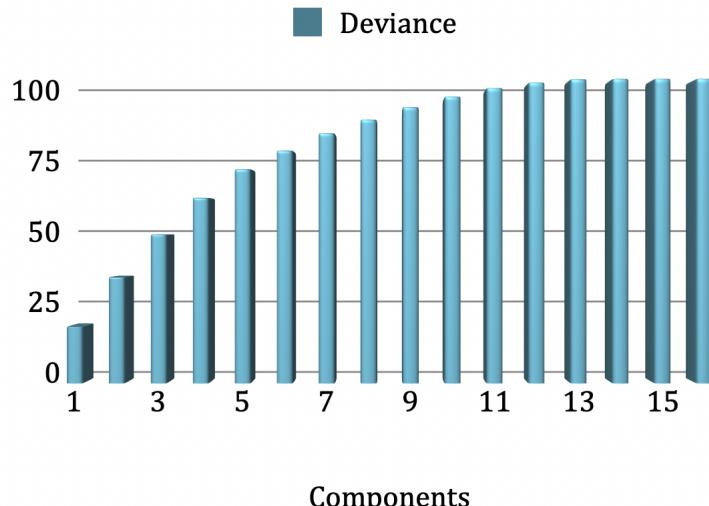


Figura 3.9: Devianza cumulativa spiegata

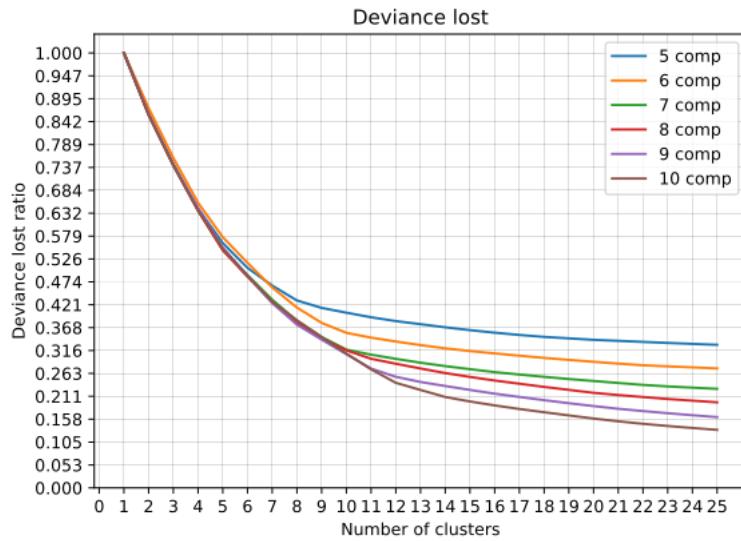


Figura 3.10: Analisi devianza

In Figura 3.10 è possibile osservare l'andamento della devianza persa in funzione del numero di cluster, al variare delle componenti principali. Con i parametri scelti (**10 componenti principali e 20 cluster**) viene preservato circa l'85% di devianza.

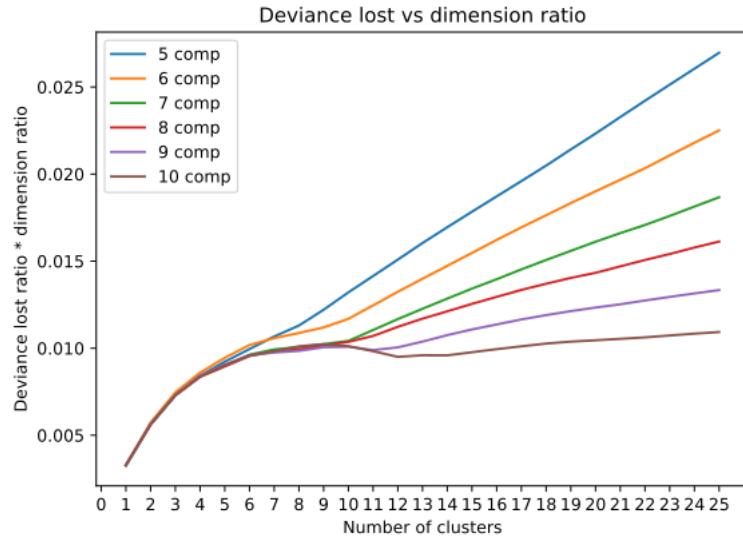


Figura 3.11: Analisi devianza

DEV_TOT	DEV_POST_PCA	DEV_POST_PCA_CL	DEV_LOST
4864	95%	85.1%	14.9%

Tabella 3.8: Analisi devianze

Infine, è stato costruito il workload sintetico, estraendo un elemento da ognuno dei cluster. Anche in questo caso l'estrazione è stata del tutto casuale.

3.2.3 Fase3

Validazione del workload sintetico di alto livello

Per validare il workload sintetico di alto livello (HL_c) ottenuto a partire da quello reale (HL), sono stati confrontati i due workload sintetici di basso livello, uno (LL_c) relativo all'applicazione del workload reale di alto livello, l'altro (LL'_c) relativo all'applicazione di quello sintetico di alto livello. Il confronto di LL_c e LL'_c è stato effettuato mediante un test di ipotesi sulla differenza delle medie.

CAPITOLO 3. WEB SERVER

Per prima cosa, per entrambi i campioni è stata verificata la normalità della distribuzione dei valori di ogni componente principale per capire se applicare un test parametrico o uno non parametrico.

A tal fine, tramite JMP per ogni componente è stato effettuato un test visivo, tracciando il QQ-plot ed è stato effettuato il test di Shapiro-Wilk per un ulteriore certezza.

Nelle figure sottostanti sono riportati i test di normalità effettuati per le prime due componenti del workload reale e di quello sintetico.

CAPITOLO 3. WEB SERVER

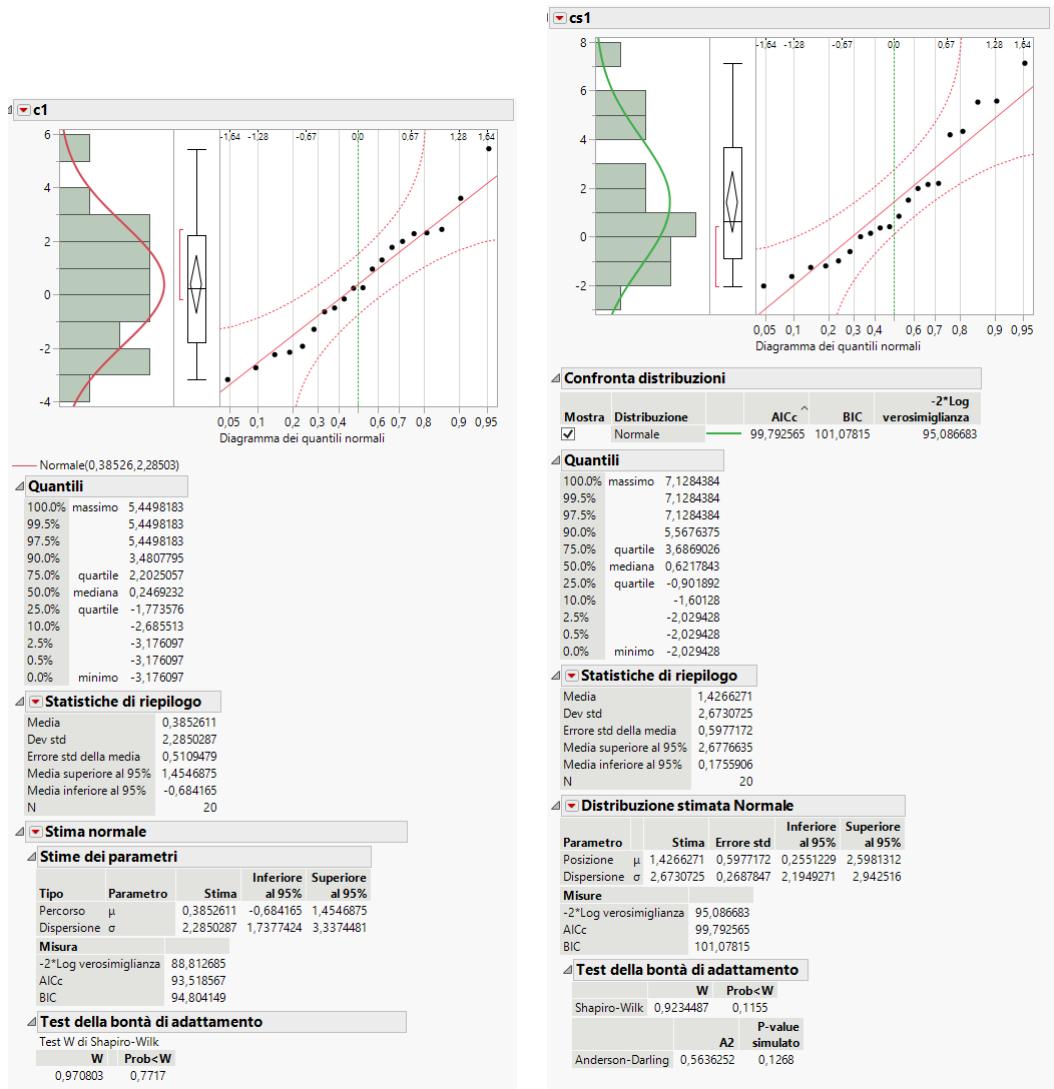


Figura 3.12: Test di normalità componente 1 (workload reale e sintetico)

CAPITOLO 3. WEB SERVER

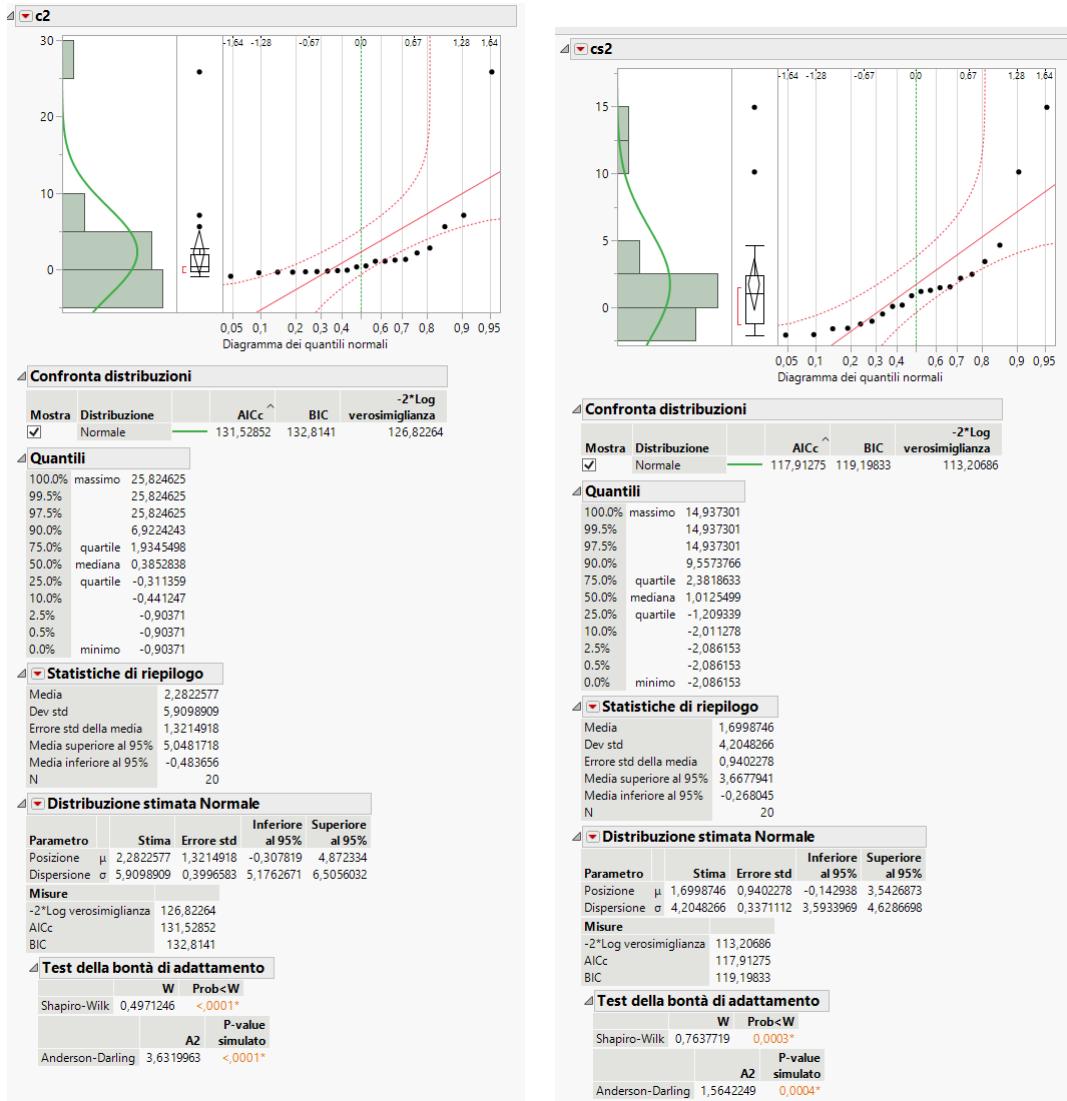


Figura 3.13: Test di normalità componente 2 (workload reale e sintetico)

É possibile vedere che, nel caso della prima componente, si ha la normalità. Al contrario, per la seconda componente, non si ha la stessa proprietà. Infatti, non è stata respinta l'ipotesi nulla del test di Shapiro-Wilk effettuato.

In definitiva, analizzando i risultati è stato riscontrato che solo le componenti 1,5 e 8 sono normali. Per quest'ultime, dato che le distribuzioni sono statisticamente

la colonna c1 (LLC) deve avere la stessa varianza di cs1 (LLC')

CAPITOLO 3. WEB SERVER

mente indipendenti, è stato applicato il test parametrico **ttest2**. Considerando che una delle assunzioni di tale test è che le distribuzioni siano caratterizzate dalla stessa varianza, è stata verificata tale caratteristica per ogni coppia di componenti utilizzando sia un test visivo, tracciando lo scatter plot dei residui rispetto alla risposta predetta che il test **vartest2**.

Nelle figure sottostanti sono mostrati i risultati di tale verifica.

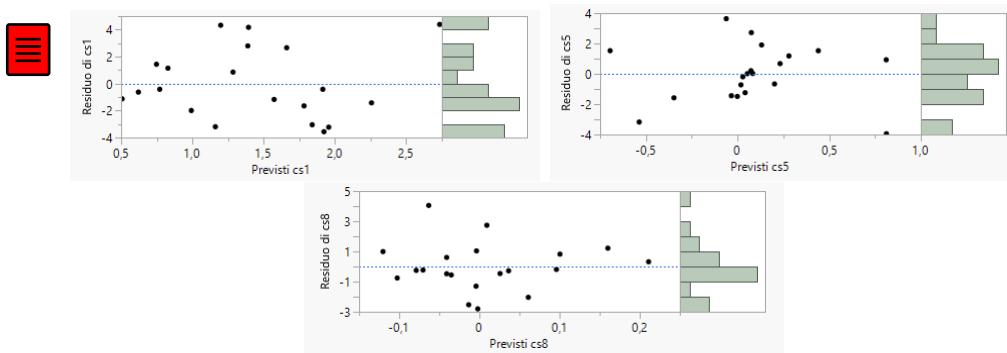


Figura 3.14: Scatter plot Residui vs Risposta prevista

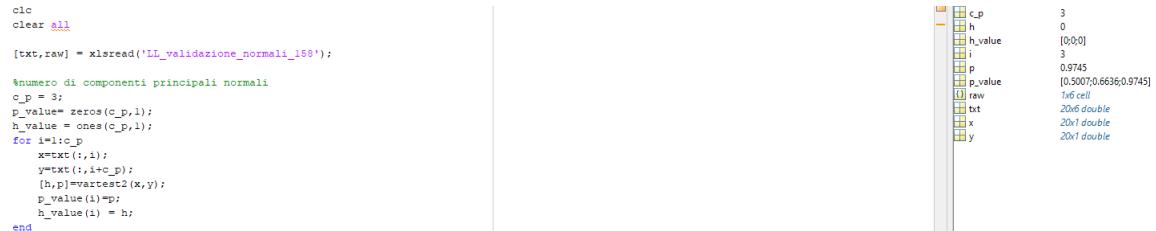


Figura 3.15: vartest2 Matlab

Per quanto riguarda il test visivo dato che non c'è la presenza di trend, possiamo assumere che le varianze siano uguali.

Allo stesso modo, dato che **vartest2** ha restituito p-value maggiori del livello di significatività fissato ($\alpha = 0.05$), l'ipotesi nulla è stata accettata in tutti i casi e le componenti presentano la stessa varianza. Quindi è stato utilizzato il **ttest2** assu-

mendo che i due campioni provengano da **distribuzioni normali con varianze uguali.**

In figura 3.16 sono mostrati i risultati di tale test.

```

clc
clear all

[txt,raw] = xlsread('LL_validazione_normali_158');

%numero di componenti principali normali
c_p = 3;
p_value= zeros(c_p,1);
h_value = ones(c_p,1);
for i=1:c_p
    x=txt(:,i);
    y=txt(:,i+c_p);
    [h,p]=ttest2(x,y);
    p_value(i)=p;
    h_value(i) = h;
end

```

Variable	Type	Value
c_p	3	
h	0	
h_value	[0;0;0]	
i	3	
p	0.3184	
p_value	[0.1933;0.5917;0.3184]	
raw	1x6 cell	
txt	20x6 double	
x	20x1 double	
y	20x1 double	

Figura 3.16: ttest2 Matlab

Per tutte le colonne considerate, il test ha restituito p-value maggiori del livello di significatività fissato ($\alpha = 0.05$), l'ipotesi nulla è stata accettata in tutti i casi e i dati provengono dalla stessa distribuzione.

Dal momento che le altre componenti non sono normali, si è dovuto optare per un test di ipotesi non parametrico. Nello specifico, è stato applicato **il test dei ranghi con segno di Wilcoxon.**

In figura 3.17 sono mostrati i risultati di tale test.

```

clc
clear all

[txt,raw] = xlsread('LL_validazione');

%numero di componenti principali
c_p = 7;
p_value= zeros(c_p,1);
h_value = ones(c_p,1);
for i=1:c_p
    x=txt(:,i);
    y=txt(:,i+c_p);
    [p,h]=ranksum(x,y);
    p_value(i)=p;
    h_value(i) = h;
end

```

Variable	Type	Value
c_p	7	
h	0	
h_value	[0;0;0;0;0;0;0]	
i	7	
p	0.0031	
p_value	[0.0101;0.0720;0.3793;...]	
raw	1x14 cell	
txt	20x14 double	
x	20x1 double	
y	20x1 double	

Figura 3.17: Test dei ranghi con segno di Wilcoxon Matlab

Anche in questo caso, per tutte le colonne considerate, il test ha restituito p-value maggiori del livello di significatività fissato ($\alpha = 0.05$), per cui l'ipotesi nulla è stata accettata in tutti i casi e i dati provengono dalla stessa distribuzione.

In conclusione, è possibile affermare che non esistono differenze significative tra i due workload sintetici di basso livello, per cui la caratterizzazione del workload di alto livello può essere ritenuta corretta.

3.3 Design of Experiment

L'ultima fase dell'analisi condotta sul web server è il **Design of Experiment**, il cui scopo è quello di determinare l'effetto di alcuni fattori sui *tempi di risposta del sistema*. Nello specifico, sono stati considerati 2 fattori, ognuno dei quali presenta un diverso numero di livelli:

- **Tasso di richieste** effettuate dal client. In questo caso, i possibili valori del fattore sono stati definiti come percentuale della usable capacity determinata in precedenza tramite il capacity test:
 - *Low intensity*: 25% della usable capacity (2500 req/s)
 - *High intensity*: 75% della usable capacity (7500 req/s)
- **Tipo di risorsa** richiesta dal client.
 - *Small size page*
 - *Small-medium size page*
 - *Medium-large size page*
 - *Large size page*

Considerando che il numero di possibili combinazioni dei valori assunti dai fattori ragionevole, si è scelto di adottare un **two-factor full-factorial design**,

così da valutare l'effetto di ogni combinazione dei livelli dei fattori sui tempi di risposta. Inoltre, per ogni combinazione sono state eseguite 5 ripetizioni, per un totale di $8 \cdot 5 = 40$ esperimenti.

3.3.1 Effetti dei fattori

Per prima cosa è necessario definire il modello della risposta del sistema.

$$y_{ijk} = \mu + \alpha_j + \beta_i + \gamma_{ij} + e_{ijk} \quad (3.1)$$

dove:

- μ è la media della risposta su tutti gli esperimenti eseguiti;
- α_j è l'effetto del livello j del primo fattore, ovvero del tasso di richiesta;
- β_i è l'effetto del livello i del secondo fattore, ovvero del tipo di risorsa;
- γ_{ij} è l'effetto dell'interazione dei livelli i e j dei due fattori;
- e_{ijk} è l'errore commesso dal modello.

Definito il modello, è possibile determinare gli effetti dei livelli di ciascun fattore come segue:

$$\alpha_j = \bar{y}_{.j.} - \bar{y}_{...}$$

$$\beta_i = \bar{y}_{i..} - \bar{y}_{...}$$

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

dove:

- $\bar{y}_{.j}$ è la media della risposta ottenuta fissando il livello j del primo fattore (tasso di richiesta) e considerando tutte le ripetizioni di tutti i livelli del secondo fattore (tipo di risorsa).
- $\bar{y}_{i..}$ è la media della risposta ottenuta fissando il livello i del secondo fattore e considerando tutte le ripetizioni di tutti i livelli del primo fattore.
- $\bar{y}_{ij.}$ è la media della risposta ottenuta fissando i livello j e i rispettivamente di primo e secondo fattore e considerando tutte le ripetizioni.
- $\bar{y}_{...}$ è la media della risposta su tutti gli esperimenti eseguiti.

Di seguito vengono riportati i risultati ottenuti:

	<i>Low</i>	<i>High</i>	Interaction1	Interaction2	Raw sum	Raw mean	Raw effect
<i>Small</i>	18	423	3433	-3433	442	221	-6760
<i>Small-med</i>	1481	1746	3503	-3503	3228	1614	-5368
<i>Med-large</i>	4696	14018	-1024	1024	18715	9357	2375
<i>Large</i>	7186	26284	-5912	5912	33471	16735	9753
Col sum	13383	42473					
Col mean	3345	10618				Total mean	
Col effect	-3636	3636				6982	



Tabella 3.9: Effetto dei fattori

3.3.2 Errori

A partire dalla risposta stimata \bar{y}_{ij} e dalla risposta osservata \hat{y}_{ijk} è possibile calcolare l'errore commesso nella stima come:

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

3.3.3 Importanza

Noti gli effetti dei fattori e gli errori commessi è possibile calcolare l'importanza dei fattori, dell'interazione e dell'errore. Si definisce importanza di un fattore (o interazione o errore) il rapporto tra la devianza del fattore (o interazione o errore) e la devianza totale. Le varianze utili per il calcolo dell'importanza si calcolano come:

$$\begin{aligned}\mathbf{SST} &= \sum_{i=1}^2 \sum_{j=1}^4 \sum_{k=1}^5 (y_{ijk} - \bar{y}_{...}) \\ \mathbf{SSA} &= 4 * 5 * \sum_{j=1}^4 (\alpha_j^2) \\ \mathbf{SSB} &= 2 * 5 * \sum_{i=1}^2 (\beta_i^2) \\ \mathbf{SSAB} &= 5 * \sum_{i=1}^2 \sum_{j=1}^4 (\gamma_{ij}^2) \\ \mathbf{SSE} &= \sum_{i=1}^2 \sum_{j=1}^4 \sum_{k=1}^5 (e_{ijk}^2)\end{aligned}$$

Ottenute le devianze si calcola l'importanza dei fattori, dell'interazione e dell'errore:

	SSA	SSB	SSAB	SSE	SST
Val	528865497,3	1753021880	600767761,1	192727425	3075382563
Percentuale	17,19%	57%	19,56%	6,26%	100%

Tabella 3.10: Devianze

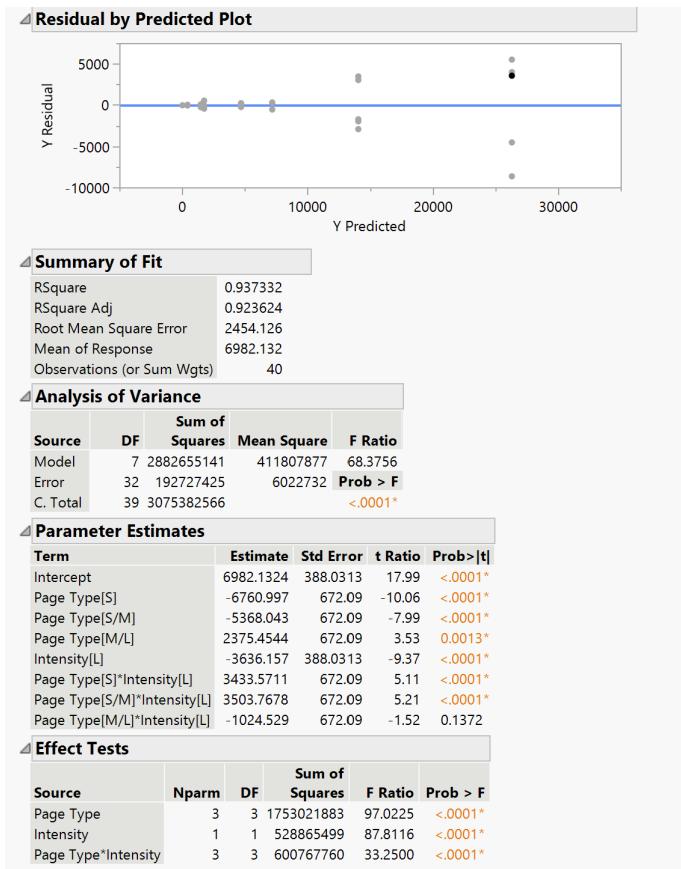


Figura 3.18: Analisi varianza

In conclusione, è possibile affermare che fattori e interazione sono notevolmente più importanti dell'errore. Inoltre, il secondo fattore, relativo al tipo di risorsa, risulta essere il più importante.

3.3.4 Significatività

L'ultimo aspetto da valutare è la significatività dei fattori, la quale rappresenta la percentuale di variabilità spiegata da un fattore rispetto a quella spiegata dall'errore. Per studiare la significatività dei fattori considerati si applica un metodo detto ANOVA (ANalysis Of VAriance). Tuttavia, il metodo preciso da applicare

CAPITOLO 3. WEB SERVER

dipende da alcune caratteristiche della distribuzione degli errori, come indicato in Tabella 3.11:

Normalità	Omoschedasticità	ANOVA
verificata	verificata	F-test
non verificata	verificata	Kruskal-Wallis test
verificata	non verificata	Welch's test
non verificata	non verificata	Kruskal-Wallis o Friedman test

Tabella 3.11: ANOVA

Per capire quale metodo applicare è necessario quindi **verificare la normalità e l'omoschedasticità** degli errori. Per verificare la **normalità** degli errori è possibile effettuare sia un test visuale, QQplot, che un test analitico, test di Shapiro-Wilk, entrambi riportati in figura 3.19. I risultati di entrambi i test **rigettano l'ipotesi che gli errori provengano da una popolazione con distribuzione normale.**

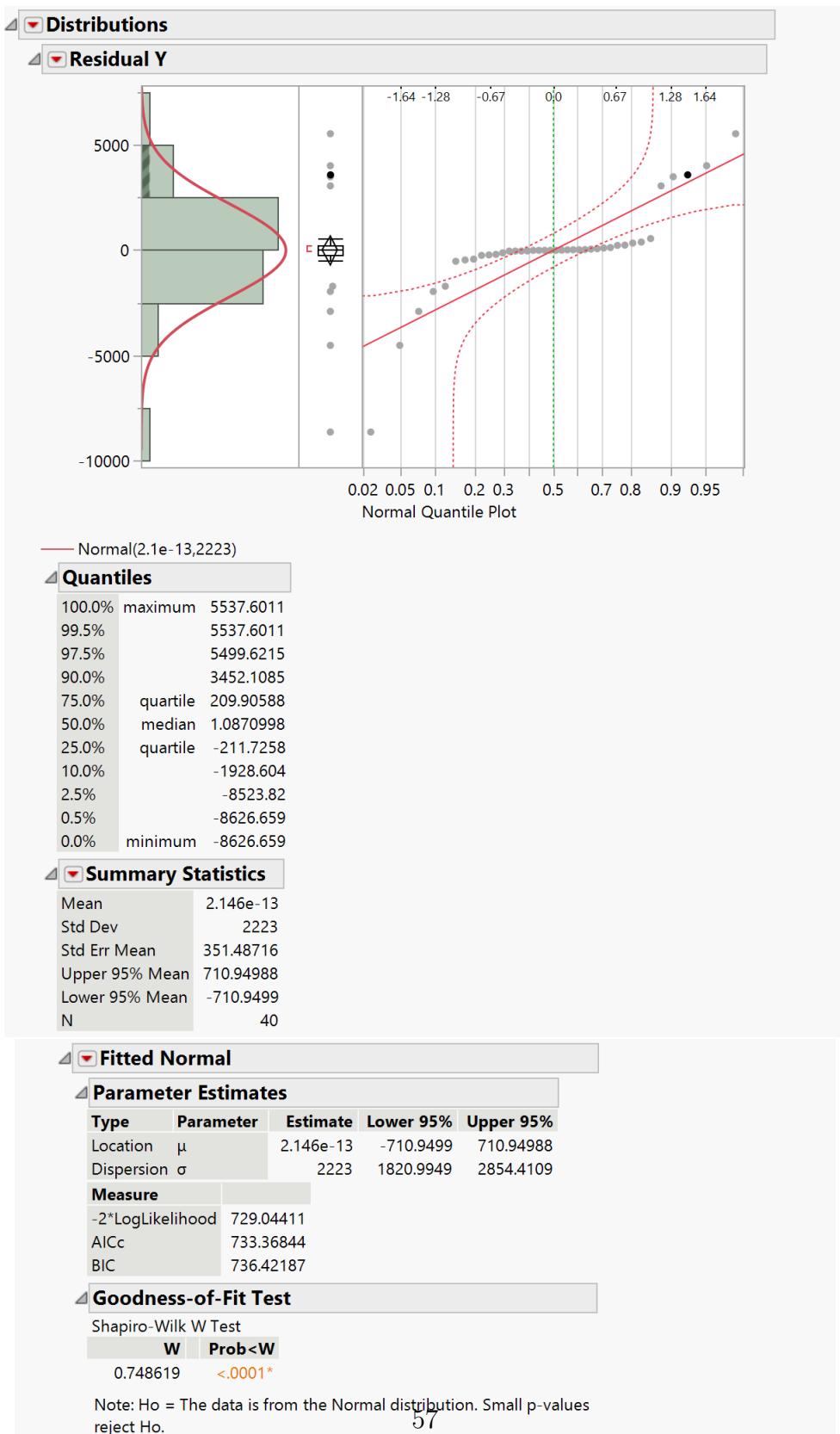


Figura 3.19: Test di normalità degli errori

Per verificare l'omoschedasticità degli errori è possibile effettuare un test visivo, acciando uno **scatter plot risposta stimata-errore**. Come mostrato in Figura 3.20, la dispersione dei punti dello scatter plot non è costante, per cui è possibile rifiutare l'ipotesi di omoschedasticità degli errori. In alternativa, è possibile effettuare un test analitico, ovvero un **test sulle varianze ineguali**, i cui risultati sono riportati in Figura 3.20. Il test rigetta l'ipotesi nulla che le varianze per ogni livello dei fattori siano uguali, per cui è possibile concludere che l'omoschedasticità degli errori non è verificata.

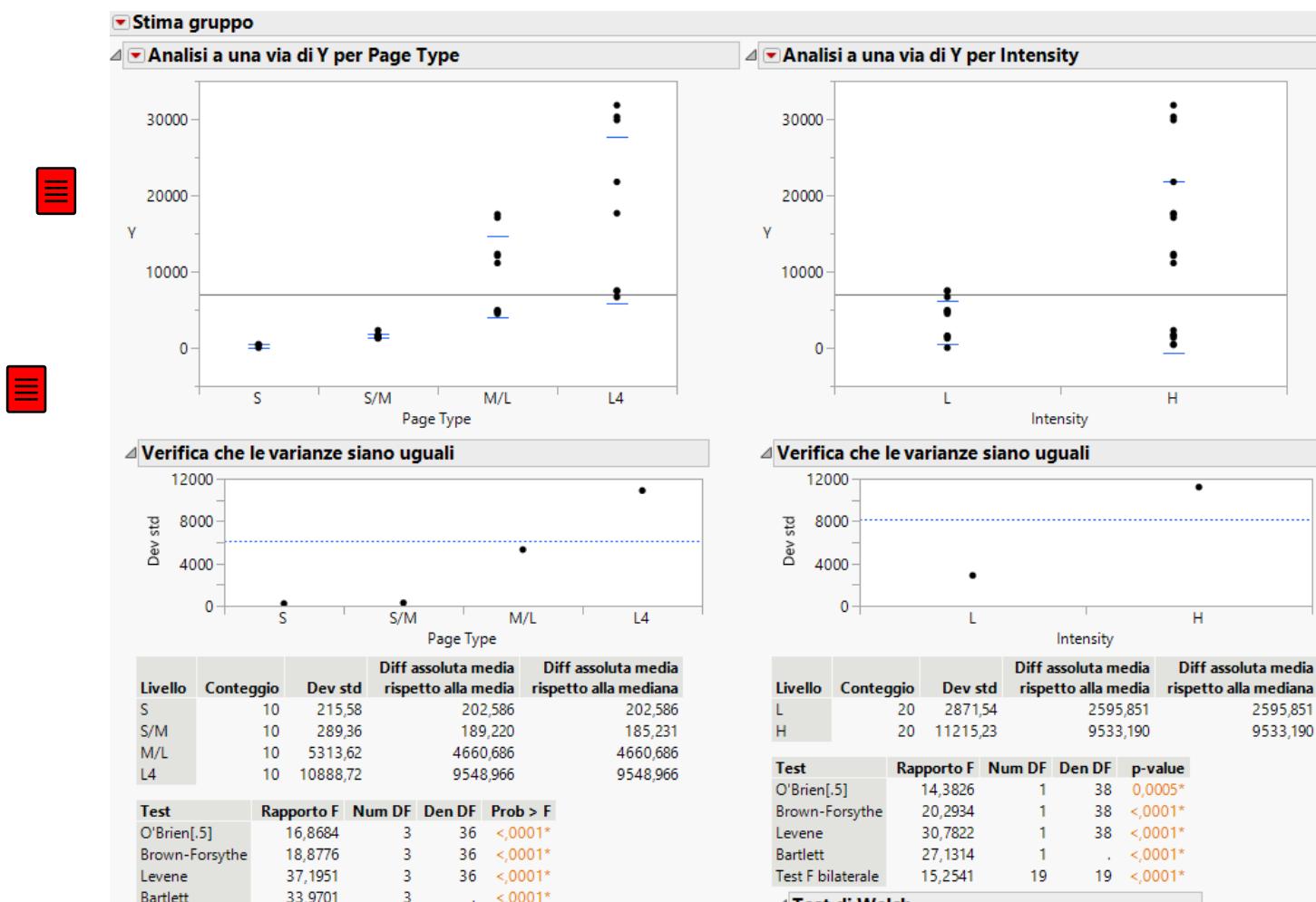


Figura 3.20: Test di omoschedasticità degli errori

Non essendo verificata né la normalità né l'omoschedasticità, si rientra nell'ultima riga della Tabella 3.11, per cui per determinare la significatività dei fattori si può effettuare un test di **Kruskal-Wallis**. Dai risultati del test riportati in Figura 3.21 emerge che solo il secondo dei due fattori, ovvero quello relativo al tipo di risorsa, è statisticamente significativo.

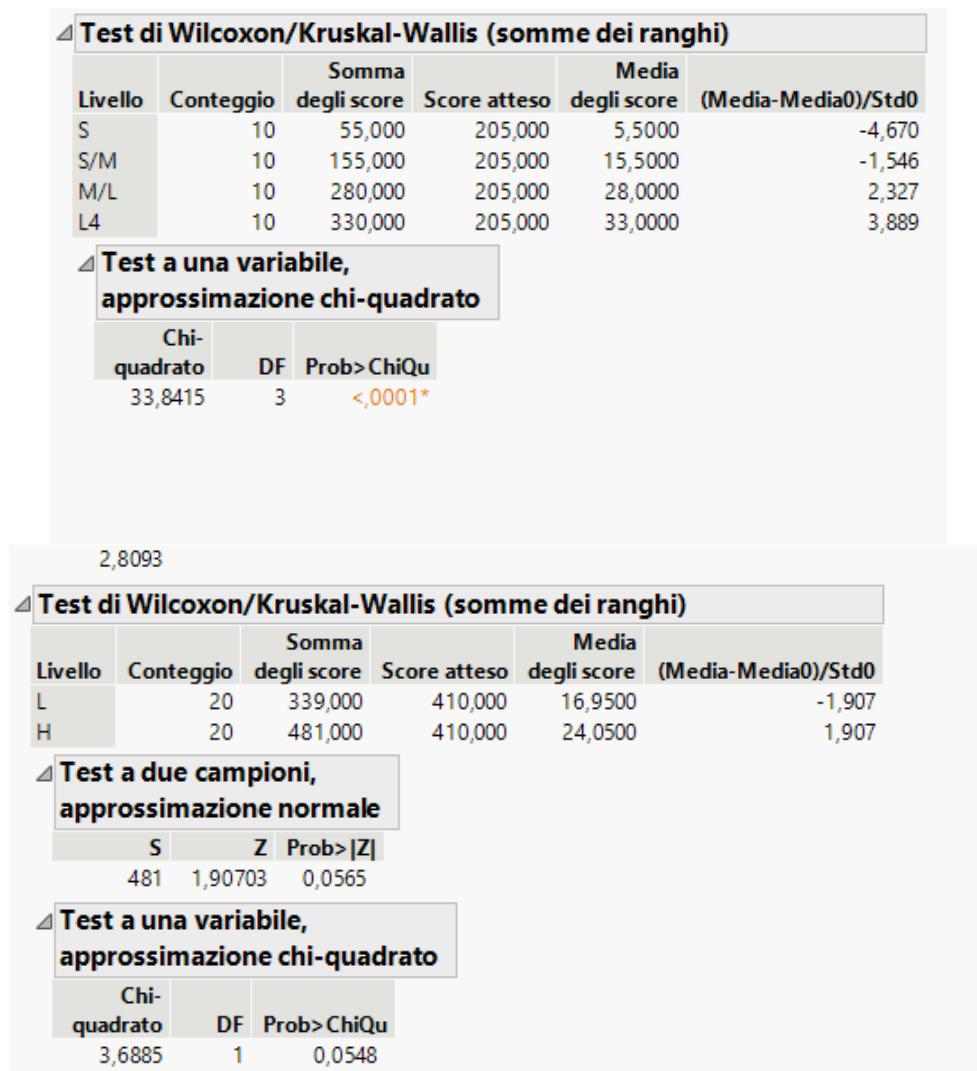


Figura 3.21: Test di significatività degli errori

Capitolo 4

Regression Model

Al fine di predire un certo errore/fenomeno su differenti distribuzioni della popolazione si è cercato di individuare un modello di regressione lineare per ogni popolazione avente a disposizione.

4.1 Descrizione del problema

La regressione lineare è un test utilizzato per individuare la presenza di trend in un certo campione. Tale campione, in un modello di regressione lineare, è composto da una coppia di valori denominati risposte e predittori. Data quindi una retta regressiva $\hat{y}_i = b_0 + b_1x_i$ candidata ad esprimere il modello, mediante lo scatter plot possiamo individuare l'errore commesso nell'approssimare un punto y_i del campione con un punto della retta, di conseguenza il primo passo per ottenere un modello di regressione lineare è scegliere tra le tante rette candidate quella che ci permette di predire in maniera ottimale l'andamento di un certo fenomeno. Tra le tante rette viene scelta, in termini di scatter plot, la retta equidistante in media da

tutti i punti, ovvero, la retta in cui la somma dei residui è nulla, successivamente ottenuto un certo numero di rette viene scelta tra le tante quella che minimizza la varianza dell'errore (SSE) e quindi massimizza la varianza totale portata nel modello regressivo. Ne possiamo dedurne che non esiste nei dati un trend lineare quando il coefficiente angolare della retta, ovvero la correlazione nei dati, è nulla o assume un valore insignificante.

4.2 Traccia 1

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

4.2.1 Exp1

Observation-Nmail

Come descritto nel paragrafo precedente il primo passo per rilevare e stimare eventuali trend è scegliere tra le possibili rette candidate quella che annulla l'errore medio e minimizza la varianza dell'errore per poi valutarne il coefficiente angolare. Mediante l'utilizzo del software JMP è possibile ottenere la retta che meglio descrive il modello predittivo.

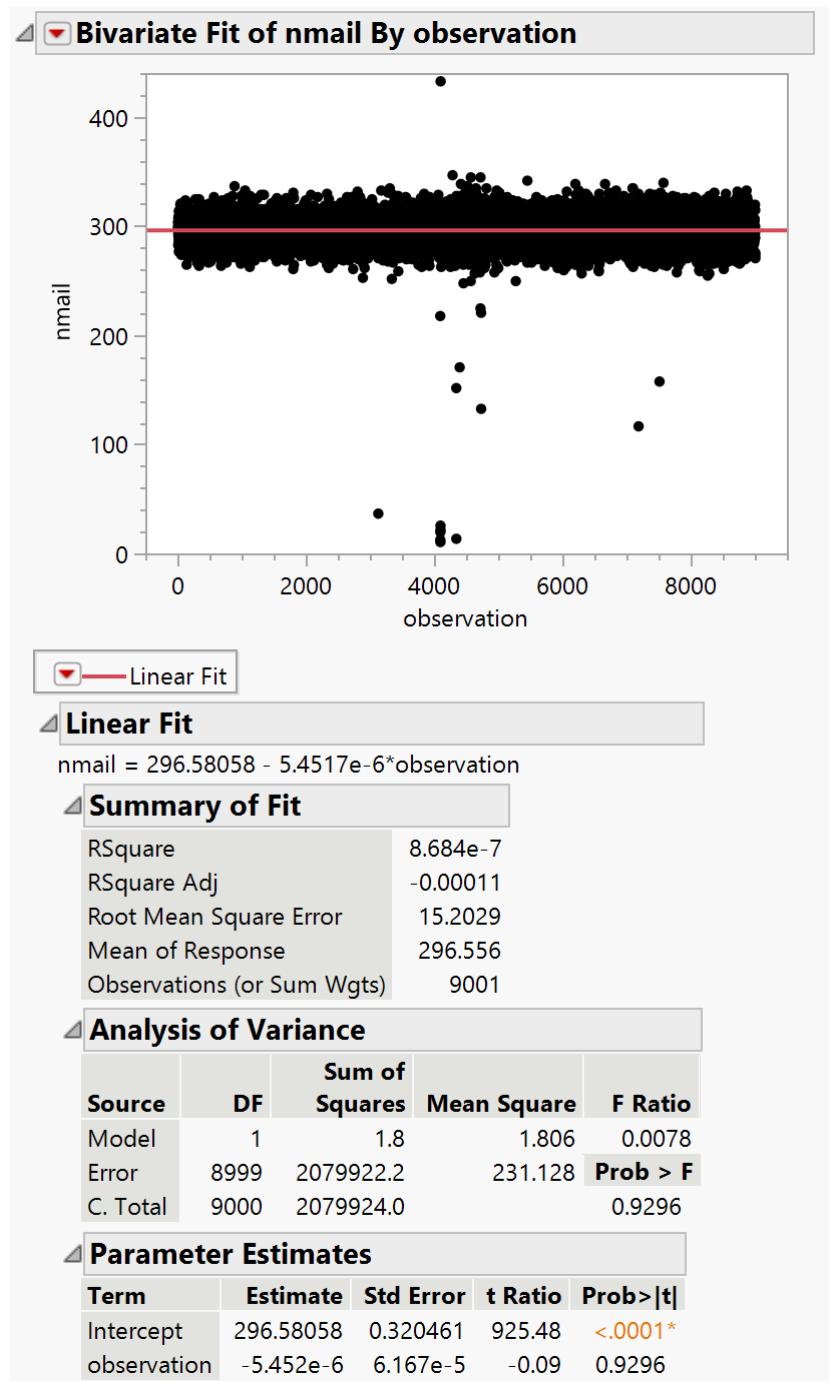


Figura 4.1: Scatter plot observation/nmail

In figura 4.1 possiamo vedere come il valore del coefficiente angolare della retta

CAPITOLO 4. REGRESSION MODEL

assume un valore insignificante, ed infatti, la retta ha un andamento piatto e non crescente, ciò indica un assenza di trend nei dati. Altri parametri che ci permettono di stimare la qualità di un modello regressivo è l' $R^2 = \frac{SSR}{SST}$, tale coefficiente indica la percentuale di variabilità spiegata dal modello regressivo, ovvero la bontà del modello regressivo stesso, in 4.1 tale valore è molto basso ciò indica che il modello individuato non è altro che un cattivo modello.

Precedentemente abbiamo detto che la retta regressiva ottenuta stima il campione, ovvero un estratto della popolazione. Tuttavia ciò che ci interessa è individuare una retta regressiva che approssimi bene la popolazione e non il campione. A tal fine occorre capire qual è l'intervallo di confidenza sia dei parametri di regressione sia della response variable. Supponendo che β_0 e β_1 siano i parametri della retta che descrive il modello regressivo della popolazione, calcolato l'intervallo di confidenza dei parametri di regressione sul campione devo capire se viene rigettata o meno l'ipotesi nulla, ovvero che l'intervallo include lo 0. Se tale ipotesi non viene rigettata allora non posso considerare il parametro β_0 differente da 0 (ciò indica un assenza di trend) al $100(1 - \alpha)\%$ di confidenza.

In 4.1 possiamo notare come il valore del *p-value* sia molto alto e che quindi l'ipotesi nulla non è rigettata, procedendo al calcolo dell'intervallo di confidenza del parametro *observation* come segue:

$$b_0 \pm ts_{b_0}$$

dove t non è altro che il $\frac{1-\alpha}{2}$ quantile della distribuzione *t-student* con $n - 2$ gradi di libertà e s_{b_0} è la deviazione standard del parametro b_0 . L'intervallo di confidenza di observation è quindi:

$$-5.452e^{-6} \pm 6.167e^{-5}$$

CAPITOLO 4. REGRESSION MODEL

Abbiamo verificato che l'ipotesi nulla non è rigettata, ovvero, lo 0 è presente all'interno dell'intervallo di confidenza.

Il modello appena utilizzato è un modello parametrico, ciò comporta che esso è basato su un insieme di assunzioni, una di queste è la presenza di relazione lineare tra la variabile di risposta e il predittore, che come abbiamo verificato precedentemente mediante uno scatter plot è assente. Di conseguenza procediamo ad utilizzare un modello non parametrico. Il test utilizzato è il *Mann-Kendal Test*, un test non parametrico la cui ipotesi nulla è l'assenza di trend nei dati. A tal fine si è implementato uno script matlab per il calcolo del τ di Kendal e l'eventuale calcolo di intercetta e coefficiente angolare mediante la *Sen's procedure*.

```
1 import pandas as pd
2 import numpy as np
3 from scipy import stats
4
5
6 sheet_name_array = ['EXP1', 'EXP2', 'os1', 'os2', 'os3', 'VMres1', '
    VMres2', 'VMres3']
7
8 xlsx = pd.ExcelFile('Homework_regressione.xlsx')
9
10
11 for i in range(0, len(sheet_name_array)):
12     df = pd.read_excel(xlsx, sheet_name=sheet_name_array[i])
13     x = df.iloc[:, 0].tolist()
14     for k in range(1, df.shape[1]):
15         y = df.iloc[:, k].tolist()
16         tau, p_value = stats.kendalltau(x=x, y=y)
17         if(p_value < 0.05):
```

```

18     slope, intercept, _, _ = stats.theilslopes(x=x,y=y)
19     print(f'datasheet name: {sheet_name_array[i]}')
20     print(f'tau: {tau}')
21     print(f'p_value: {p_value}')
22     print(f'slope: {slope}')
23     print(f'intercept: {intercept} \n')
24 else:
25     print(f'datasheet name: {sheet_name_array[i]}')
26     print(f'tau: {tau}')
27     print(f'p_value: {p_value} \n')

```

Listing 4.1: Kendall-Sen's procedure

I risultati ottenuti sono i seguenti:

τ	p-value
-0.00258	0.71638

Tabella 4.1: observaztion/nmail Kendal test

L'ipotesi nulla non è rigettata quindi anche il test di Kendall conferma l'assenza di una dipendenza lineare nei dati. Possiamo confrontare i risultati ottenuti dallo script python con i risultati ottenuti mediante JMP (Figura 4.2).

CAPITOLO 4. REGRESSION MODEL

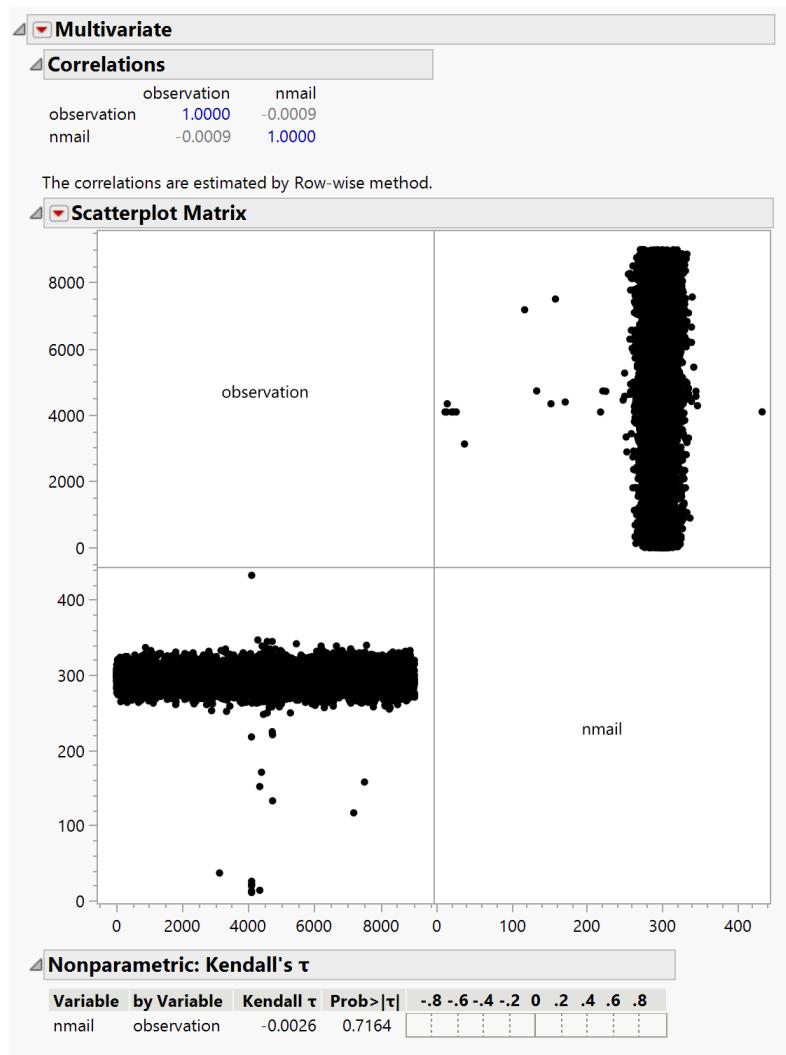


Figura 4.2: Kendal test observation/nmail

Observation-byteRec

Procediamo al calcolo della retta di regressione in JMP (Figura 4.3).



CAPITOLO 4. REGRESSION MODEL

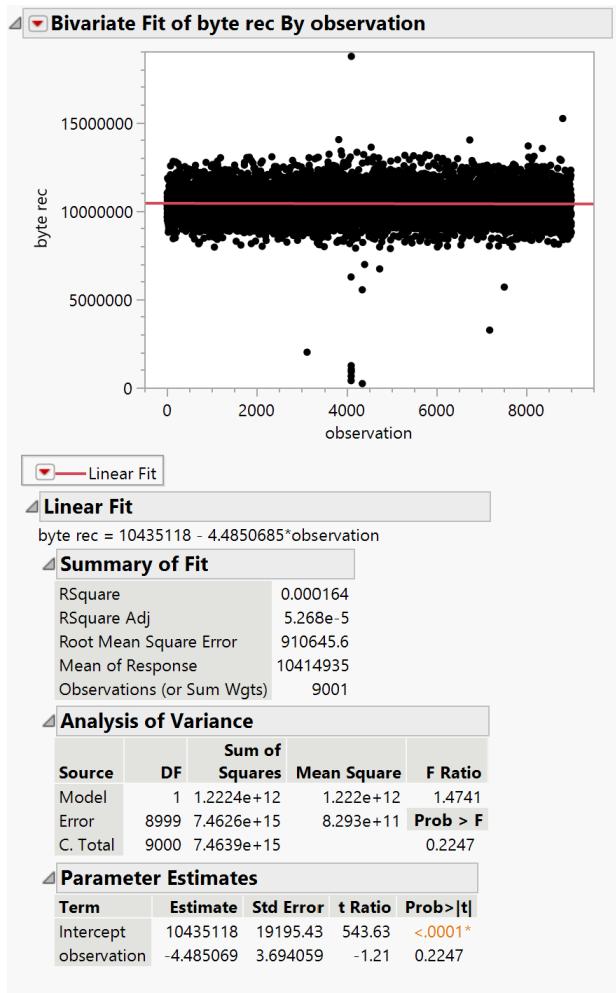


Figura 4.3: Scatter plot observation/bytRec

Come si evidenzia in 4.3 non è possibile individuare un trend in quanto il coefficiente angolare della retta di regressione assume un valore molto basso. Notiamo come l' R^2 è altrettanto basso e l'ipotesi nulle per l'intervallo di confidenza non viene rigettata. L'assenza di trend è confermata dal test di Kendall (Table 4.2).

τ	p-value
-0.00900	0.20018

Tabella 4.2: observaztion/bytRec Kendall test

Observation-byteSent

Procediamo al calcolo della retta di regressione in JMP (Figura 4.4).

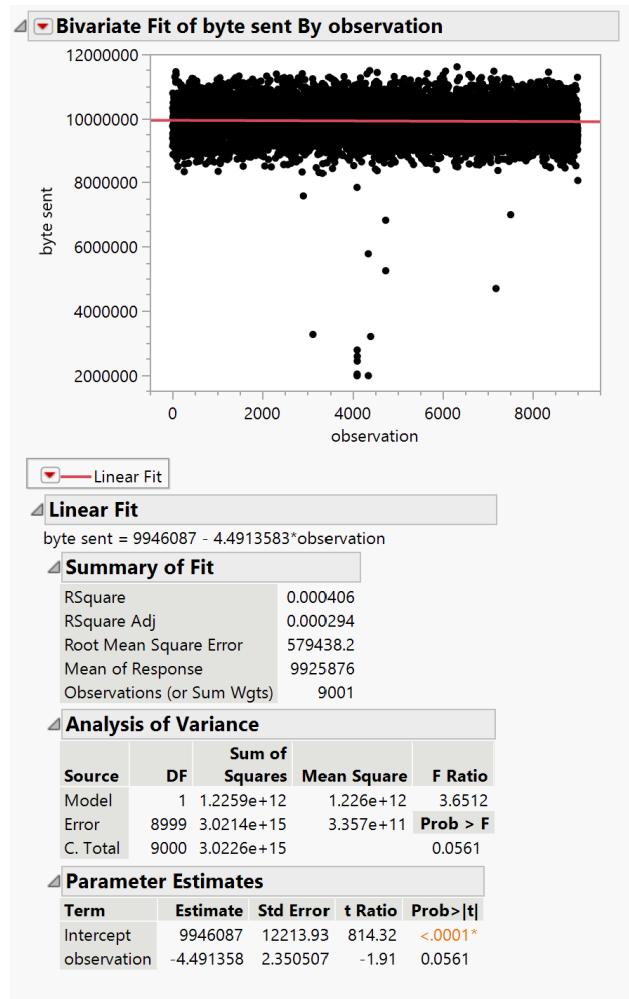


Figura 4.4: Scatter plot observation/bytesent

La figura 4.4 mostra come non sia rispettata l'ipotesi di una relazione lineare tra la variabile di risposta e il predittore. Applicando ai dati il test di *Mann Kendall* otteniamo risultati differenti come possiamo notare nella tabella 4.3.

CAPITOLO 4. REGRESSION MODEL

τ	p-value	slope	intercept
-0.01418	0.04349	-4.51478	9959247.52

Tabella 4.3: observaztion/bytēSent Kendal test

Possiamo confrontare i risultati ottenuti con il software JMP (Figura 4.5).

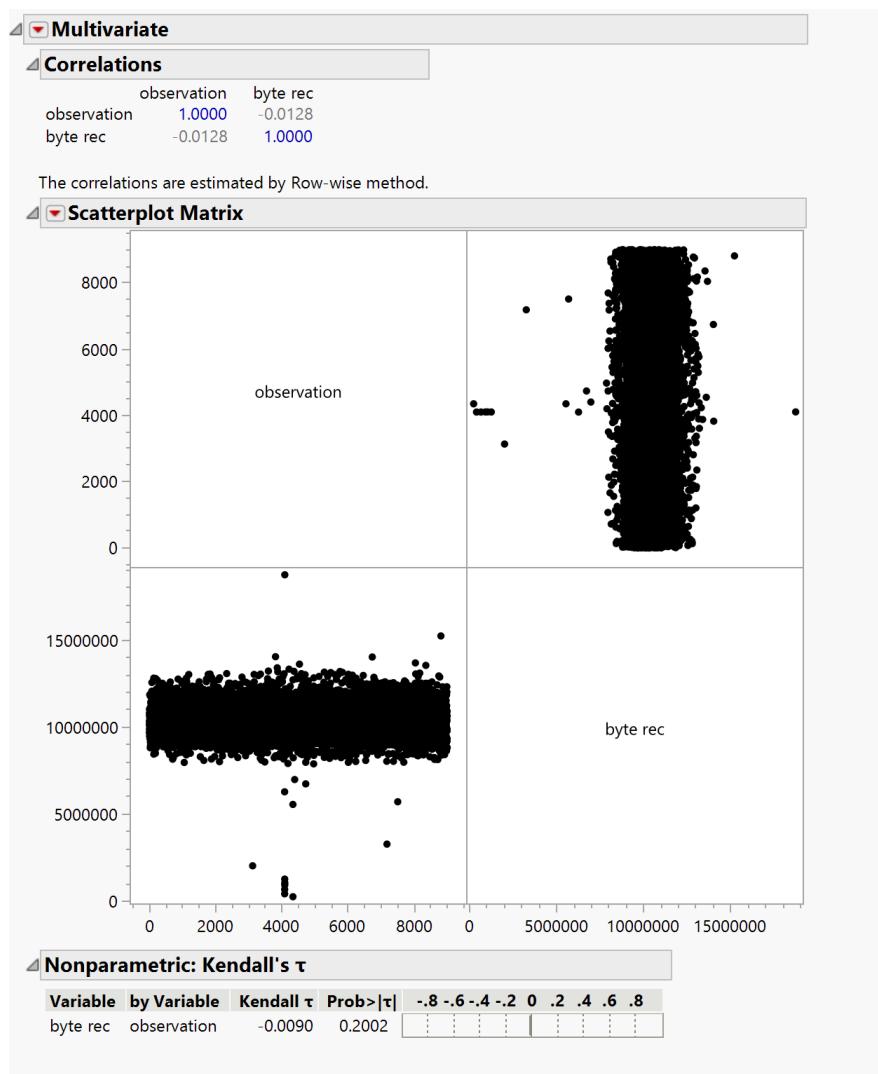


Figura 4.5: Kendal test observation/bytesent

Nei dati è quindi presente un trend in quanto il test rigetta l'ipotesi nulla.

Mediante lo script python si è calcolata l'intercetta e il coefficiente angolare della

retta che stima il modello.

4.2.2 Exp2

Observation-Nmail

Procediamo al calcolo della retta di regressione in JMP (Figura 4.6).

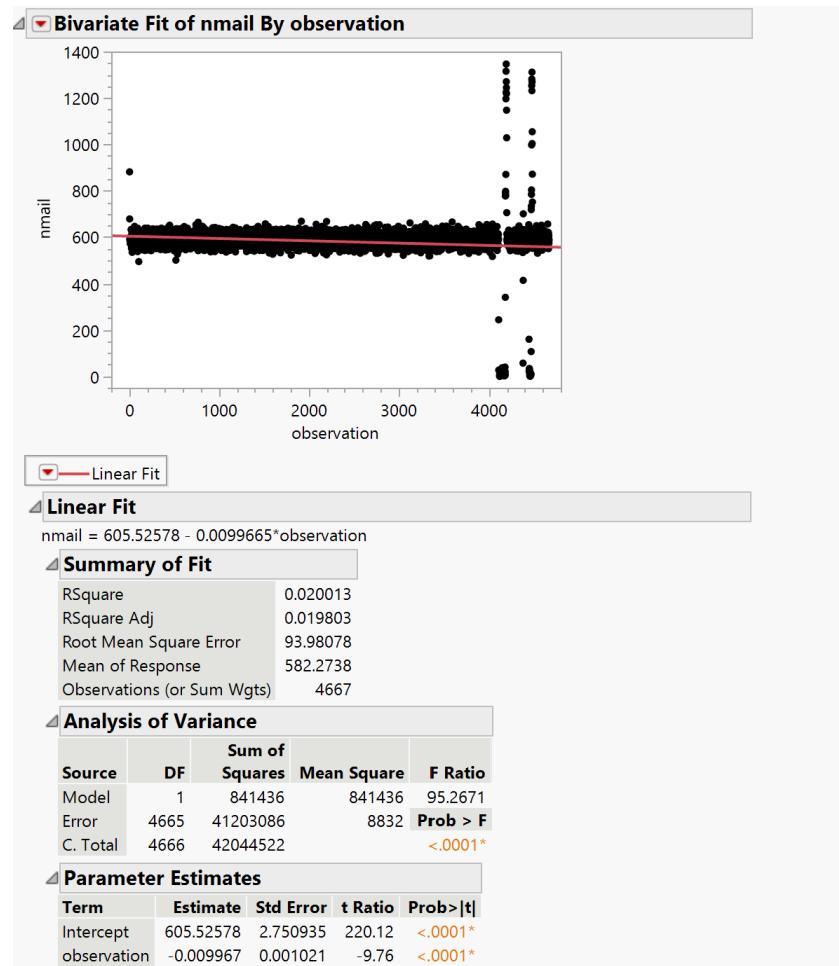


Figura 4.6: Scatter plot observation/nmail

La retta estimata presenta un valore del coefficiente angolare non nullo, inoltre il p-value è basso e di conseguenza l'ipotesi nulla (lo 0 è incluso nell'intervallo di

confidenza) è rigettata. Anche se notiamo che il valore dell' R^2 è basso, ciò significa che con questo modello portiamo poca varianza. Come già detto precedentemente il modello utilizzato è un modello parametrico e quindi basato su un insieme di assunzioni:

- È presente una relazione lineare tra la variabile di risposta e il predittore
- Gli errori devono essere indipendenti
- Gli errori devono essere distribuiti normalmente
- Gli errori devono avere deviazione standard costante

La prima assunzione come si può evincere dalla figura 4.6 è verificata. Procediamo quindi a verificare le restanti assunzioni.

Errori indipendenti Per verificare se gli errori sono indipendenti si può procedere rappresentando i residui rispetto al predittore su uno scatter plot, la presenza di trend indica una dipendenza dell'errore della variabile di predizione.

residui rispetto al valore previsto(y del dataset)

CAPITOLO 4. REGRESSION MODEL

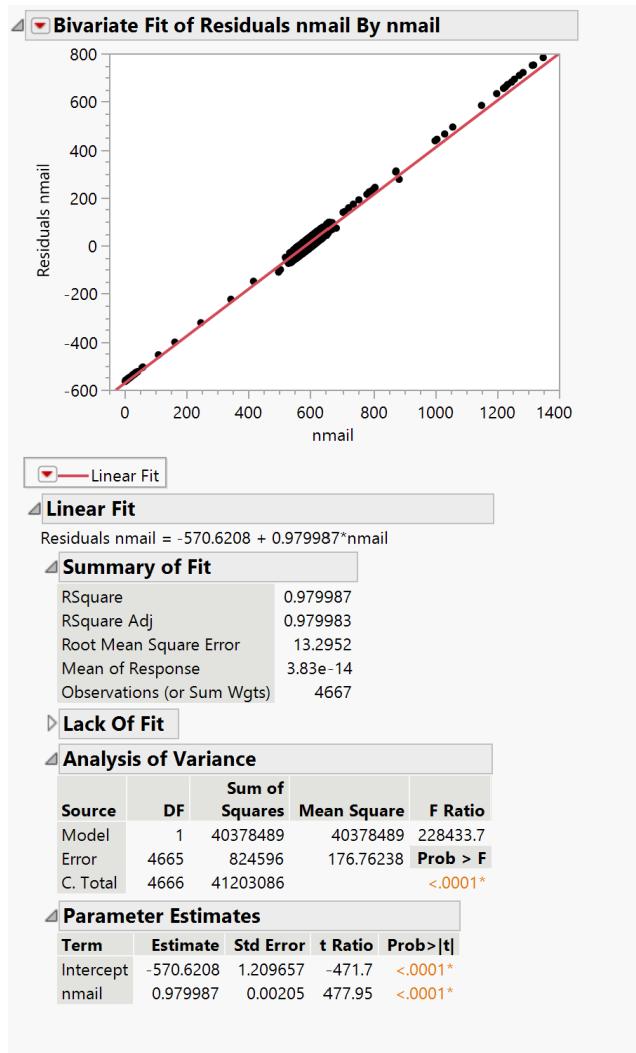


Figura 4.7: Scatter plot residui/nmail

In figura 4.7 si può vedere come il coefficiente angolare della retta non è nullo e quindi è presente una dipendenza lineare tra i residui e la variabile di predizione.

Errori distribuiti normalmente Per verificare se gli errori sono distribuiti come una normale si è utilizzato un *q-q plot* (Figura 4.8 osservando se i quantili dei residui si posizionano similmente ai quantili della normale).

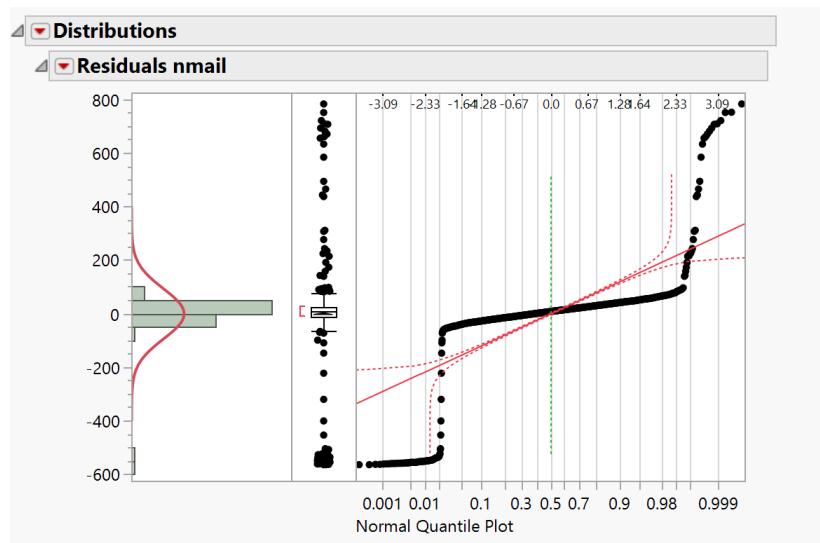


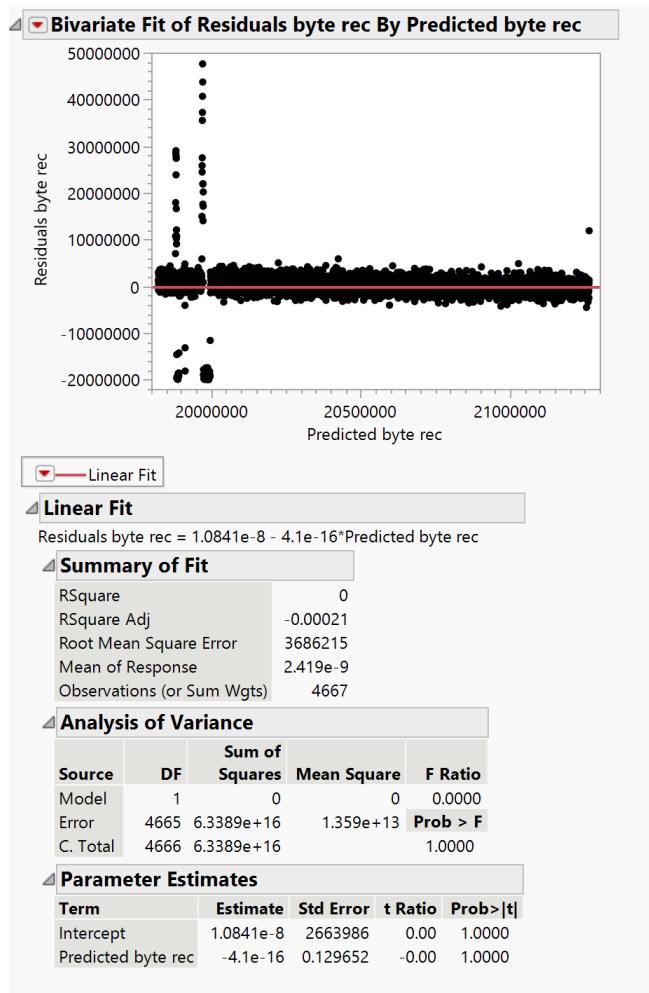
Figura 4.8: QQ plot residui

Si può notare la presenza di *short tails* da cui ne deduciamo che i residui non sono normali.

Omoschedasticità dei residui Per verificare se i residui hanno una deviazione standard costante basta osservare lo scatter plot tra l'errore e la predicted response. Se la diffusione di una parte del grafico è visivamente differente da quella di un'altra parte del grafico allora la distribuzione degli errore non è costante ma dipende dalla predicted response. Possiamo vedere in Figura 4.9 che i residui sono omoschedastici.

errore e y
predetta

CAPITOLO 4. REGRESSION MODEL



sbagliato
doveva essere
predicted nmail,
residual nmail

Figura 4.9: Scatter plot residui/predicted response

Non essendo verificate alcune delle assunzioni del modello parametrico si procede con **un modello non parametrico e quindi il test di kendall**. I risultati ottenuti sono scritti nella tabella 4.4.

τ	p-value	slope	intercept
-0.02417	0.01384	-0.0005	592.39366

Tabella 4.4: observation/nmail Kendal test

È presente quindi un trend nei dati e la retta che meglio stima il modello è

CAPITOLO 4. REGRESSION MODEL

descritta dai valori di intercetta e coefficiente angolare presenti in tabella.

Observation-byteRec

Procediamo al calcolo della retta di regressione in JMP (Figura 4.10).

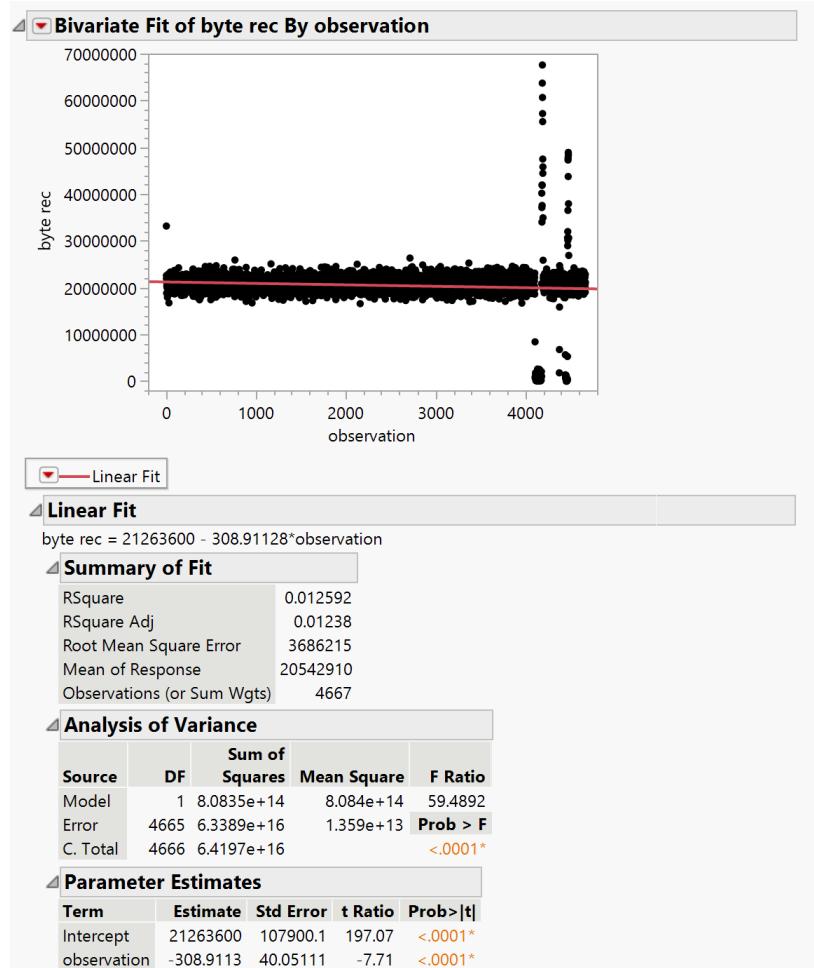


Figura 4.10: Scatter plot observation/bytRec

La retta estimata presenta un valore del coefficiente angolare non nullo, inoltre il p-value è basso e di conseguenza l'ipotesi nulla (lo 0 è incluso nell'intervallo di confidenza) è rigettata.

CAPITOLO 4. REGRESSION MODEL

Le assunzioni per il modello parametrico non sono tutte soddisfatte si procede quindi con il test di *Kendall* ottenendo i risultati in tabella 4.5

τ	p-value	slope	intercept
-0.0315	0.00121	-48.29360	20890131.991

Tabella 4.5: observation/byteRec Kendal test

Observation-byteSent

Procediamo al calcolo della retta di regressione in JMP (Figura 4.11).

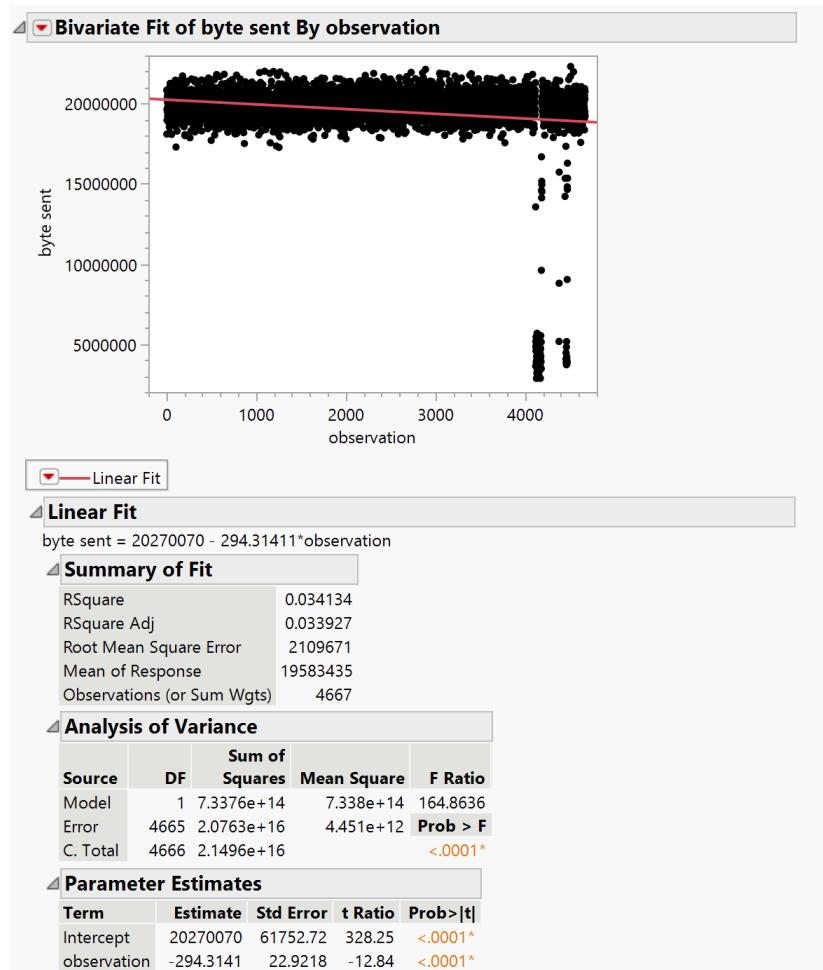


Figura 4.11: Scatter plot observation/byteSent

Solo per il primo caso abbiamo riportato le cose da controllare, qui no, ma non sono verificate

CAPITOLO 4. REGRESSION MODEL

La retta estimata presenta un valore del coefficiente angolare non nullo, inoltre il p-value è basso e di conseguenza l'ipotesi nulla (lo 0 è incluso nell'intervallo di confidenza) è rigettata.

Le assunzioni per il modello parametrico non sono tutte soddisfatte si procede quindi con il test di *Kendall* ottenendo i risultati in tabella 4.6

τ	p-value	slope	intercept
-0.03923	$5.84715e^{-5}$	-34.06305	19903064.100

Tabella 4.6: observation/byteSent Kendal test

4.3 Traccia 2

Rilevare e stimare eventuali trend sulle 5 variabili utilizzando modelli regressivi lineari semplici, parametrici e/o non parametrici. Farlo per i tre dataset os1, os2 e os3. Confrontare i trend individuati nei tre dataset.

4.3.1 Risultati os1, os2, os3

Come per la traccia 1 si è prima utilizzato un modello regressivo parametrico, verificando la veridicità dell'insieme di assunzione su cui esso è basato. Per tutti i dataset in esame e per ogni coppia *Time-variable* in essi contenute non sono rispettate tutte le assunzioni, in particolare:

- Os1
 - *LIN1 – VmSize* : normalità e omoschedasticità
 - *LIN1 – VmData* : normalità e omoschedasticità

- $LIN1 - RSS$: normalità e omoschedasticità
- $LIN1 - byte - letti - I - O$: normalità
- $LIN1 - byte - letti - I - O1$: normalità
- Os2
 - $LIN2 - VmSize$: normalità
 - $LIN2 - VmData$: normalità
 - $LIN2 - RSS$: normalità e omoschedasticità
 - $LIN2 - byte - letti - sec$: normalità
 - $LIN2 - byte - scritti - sec$: normalità
- Os3
 - $LIN4 - VmSize$ normalità
 - $LIN4 - VmData$ normalità
 - $LIN4 - RSS$ normalità
 - $LIN4 - byte - letti - sec$ normalità e dipendenza lineare
 - $LIN4 - byte - scritti - sec$ normalità e dipendenza lineare

Si è utilizzato, quindi, un modello non parametrico per la stima dei trend.

Os1

I risultati ottenuti mediante il test di *Mann Kendall* per il dataset *Os1* sono riportati nella tabella 4.7.

CAPITOLO 4. REGRESSION MODEL

	τ	p-value	slope	intercept
LIN1-VmSize	0.80764	0.0	0.03157	738854.3947
LIN1-VmData	0.80764	0.0	0.03157	670450.3947
LIN1-RSS	0.68268	0.0	0.00497	42428.8070
LIN1-byte-letti-I-O	0.08230	$5.36155e^{-18}$	4.73757	102763229.5031
LIN1-byte-letti-I-O1	-0.00173	0.85547	/	/

Tabella 4.7: Kendall test os1

Possiamo confrontare i risultati ottenuti con lo script matlab con quelli ottenuti mediante il software JMP in figura 4.12.

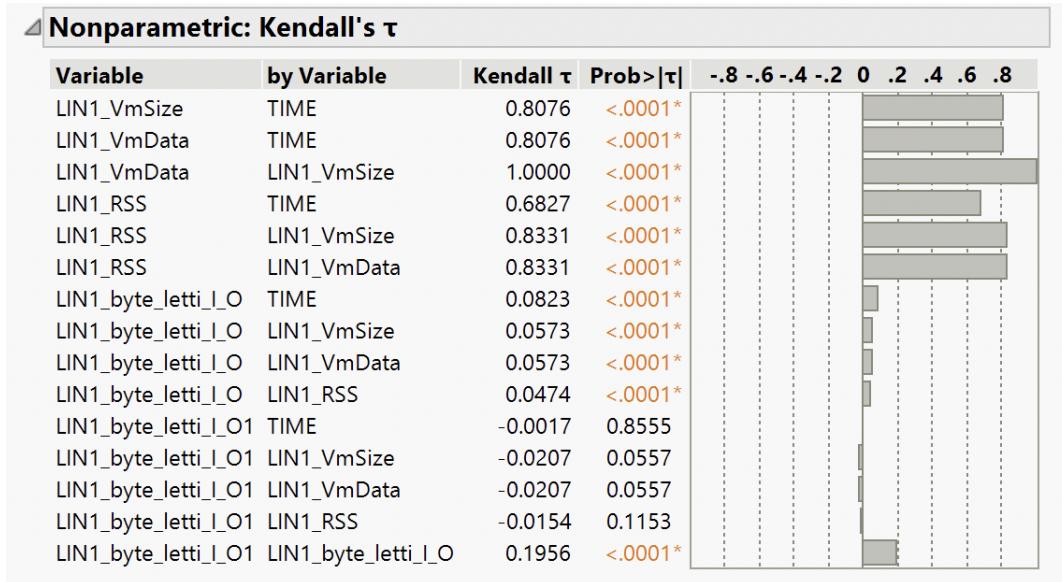


Figura 4.12: Kendall test os1

Os2

I risultati ottenuti mediante il test di *Mann Kendall* per il dataset *Os2* sono riportati nella tabella 4.8.

	τ	p-value	slope	intercept
LIN2-VmSize	0.70648	0.0	0.00191	735790.3761
LIN2-VmData	0.70648	0.0	0.00191	667414.3761
LIN2-RSS	0.17877	1.90894	0.00063	42384.8545
LIN2-byte-letti-sec	0.62594	0.0	114.55350	77733384.2378
LIN2-byte-scritti-sec	-0.02102	0.03419	-0.48861	65693615.5762

Tabella 4.8: Kendall test os1

Possiamo confrontare i risultati ottenuti con lo script matlab con quelli ottenuti mediante il software JMP in figura 4.13.

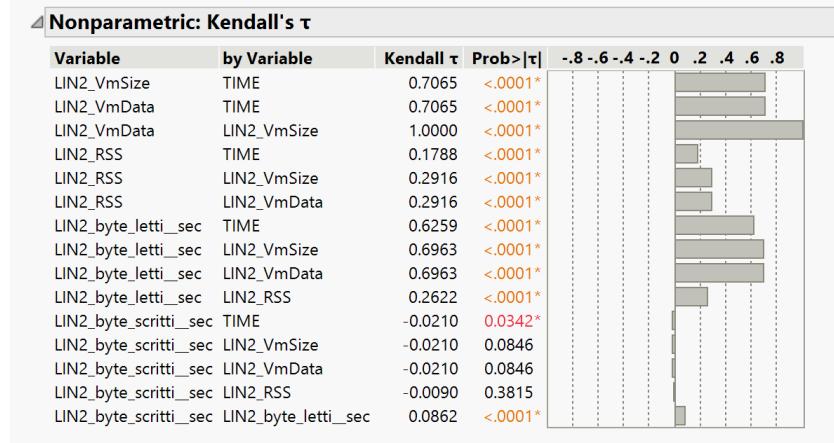


Figura 4.13: Kendall test os1

Os3

I risultati ottenuti mediante il test di *Mann Kendall* per il dataset *Os3* sono riportati nella tabella 4.9.

	τ	p-value	slope	intercept
LIN4-VmSize	0.78322	0.0	0.02165	744374.5991
LIN4-VmData	0.78322	0.0	0.02165	675970.5991
LIN4-RSS	0.59381	0.0	0.00345	45386.4710
LIN4-byte-letti-sec	-0.05055	$3.27005e^{-7}$	-1.89104	141291080.7714
LIN4-byte-scritti-sec	-0.01709	0.08412	/	/

Tabella 4.9: Kendal test os1

Possiamo confrontare i risultati ottenuti con lo script matlab con quelli ottenuti mediante il software JMP in figura 4.14.

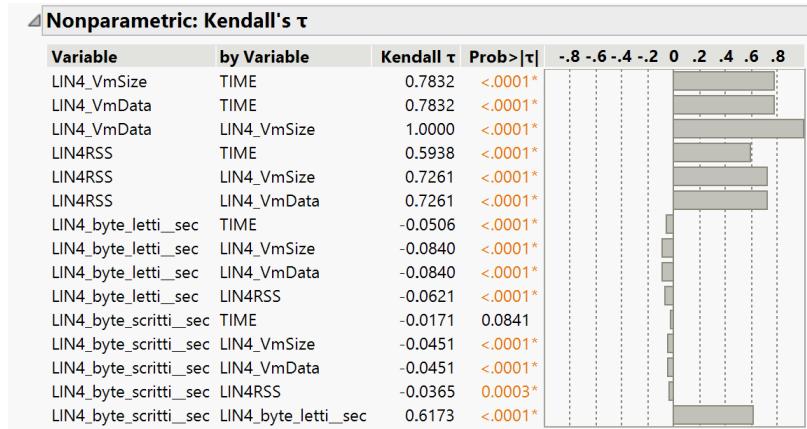


Figura 4.14: Kendal test os1

4.4 Traccia 3

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

4.4.1 Vmres1

Come nelle tracce precedenti, per rilevare un eventuale trend, si è applicato sul dataset in esame un **modello parametrico verificando la veridicità delle assunzioni mediante il software JMP**. Non rispettando tutte le assunzioni si è utilizzato JMP per verificare la presenza di un trend mediante un test non parametrico.

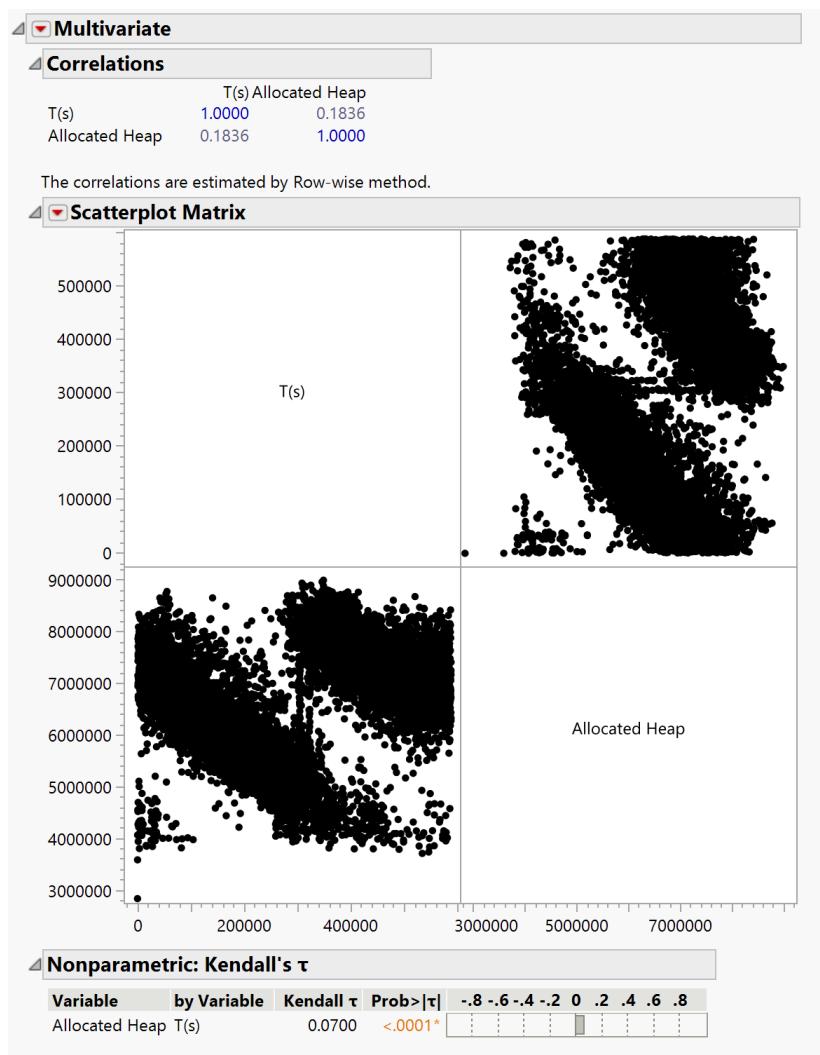


Figura 4.15: Kendall test VMRes1

Possiamo notare in figura 4.15 che il p-value ottenuto applicando ai dati il

Kendal test assume un valore piccolo, rigettando quindi l'ipotesi nulla, cioè **confirma la presenza di trend nei dati**. Successivamente mediante lo script python, il cui codice è stato riportato nella sezione 1, sono stati individuati gli elementi necessaria alla costruzione della retta di regressione, ovvero coefficiente angolare e intercetta, riportati in tabella 4.10.

τ	p-value	slope	intercept
0.07002	$2.57618e^{-25}$	0.69112	6562275.4616

Tabella 4.10: Kendal test VMres1

Data quindi l'equazione della retta: $y = slope*x + intercept$ è possibile ricavare il punto di saturazione mediante una formula inversa:

$$saturation-point = (1024^3 - intercept) / slope$$

Ottenendo che il tempo di saturazione della memoria heap sarà di **48.963885629394625g**.

4.4.2 Vmres2

Come nelle tracce precedenti, per rilevare un eventuale trend, si è applicato sul dataset in esame un modello parametrico verificando la veridicità delle assunzioni mediante il software JMP. Non rispettando tutte le assunzioni si è utilizzato JMP per verificare la presenza di un trend mediante un test non parametrico.

CAPITOLO 4. REGRESSION MODEL

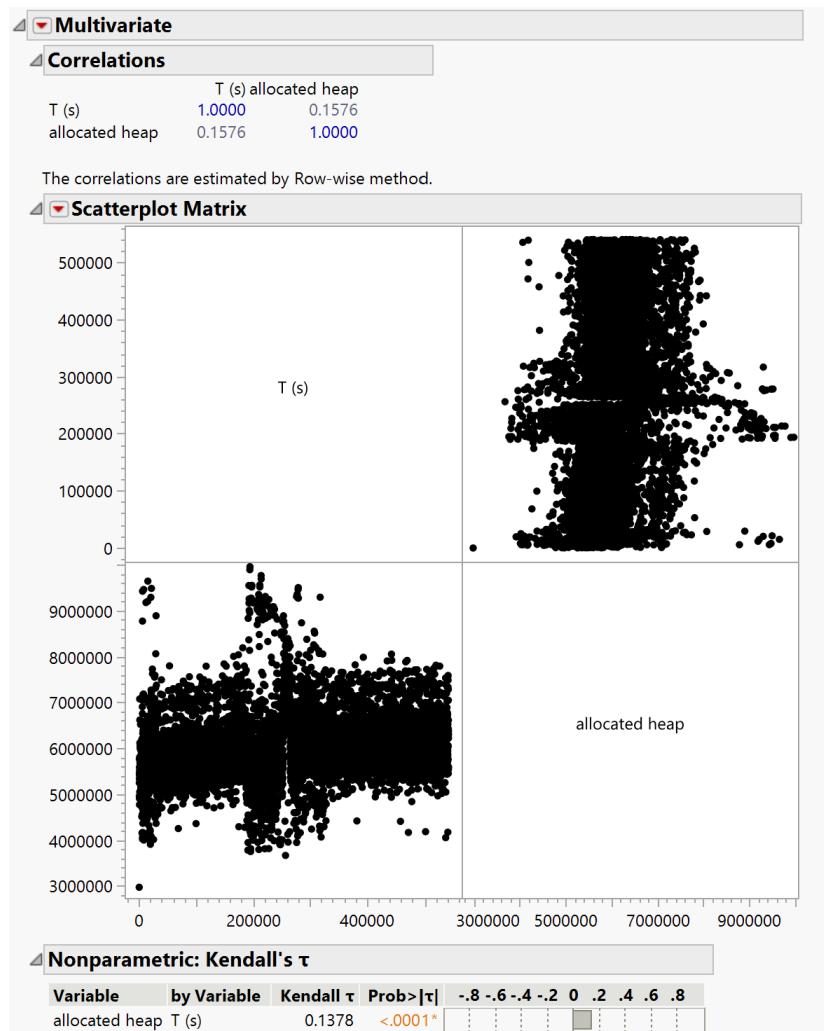


Figura 4.16: Kendal test VMres2

Possiamo notare in figura 4.16 che il p-value ottenuto applicando ai dati il *Kendal* test assume un valore piccolo, rigettando quindi l'ipotesi nulla, ciò conferma la presenza di trend nei dati. Successivamente mediante lo script python, il cui codice è stato riportato nella sezione 1, sono stati individuati gli elementi necessaria alla costruzione della retta di regressione, ovvero coefficiente angolare e intercetta, riportati in tabella 4.11.

τ	p-value	slope	intercept
0.13781	7.96638^{-86}	0.67094	5780180.5633

Tabella 4.11: Kendal test VMres2

Data quindi l'equazione della retta: $y = slope*x + intercept$ è possibile ricavare il punto di saturazione mediante una formula inversa:

$$saturation-point = (1024^3 - intercept)/slope$$

Ottenendo che il tempo di saturazione della memoria heap sarà di 50.47297871681608g.

4.4.3 Vmres3

Come nelle tracce precedenti, per rilevare un eventuale trend, si è applicato sul dataset in esame un modello parametrico verificando la veridicità delle assunzioni mediante il software JMP. Non rispettando tutte le assunzioni si è utilizzato JMP per verificare la presenza di un trend mediante un test non parametrico.

Possiamo notare in figura 4.17 che il p-value ottenuto applicando ai dati il *Kendal* test assume un valore piccolo, rigettando quindi l'ipotesi nulla, ciò conferma la presenza di trend nei dati.

CAPITOLO 4. REGRESSION MODEL

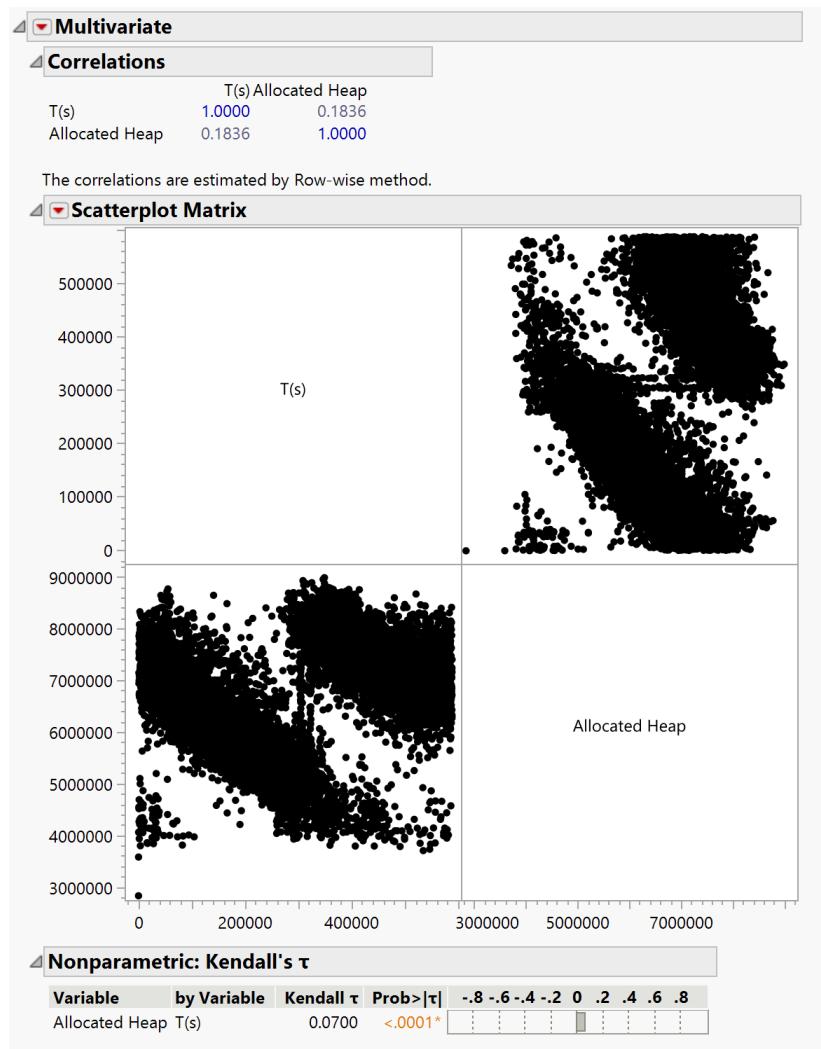


Figura 4.17: Kendal test VMres3

Successivamente mediante lo script python, il cui codice è stato riportato nella sezione 1, sono stati individuati gli elementi necessaria alla costruzione della retta di regressione, ovvero coefficiente angolare e intercetta, riportati in tabella 4.12.

τ	p-value	slope	intercept
0.20309	2.08339^{-185}	2.90309	6041585.5564

Tabella 4.12: Kendal test VMres3

CAPITOLO 4. REGRESSION MODEL

Data quindi l'equazione della retta: $y = slope*x + intercept$ è possibile ricavare il punto di saturazione mediante una formula inversa:

$$saturation-point = (1024^3 - intercept) / slope$$

Ottenendo che il tempo di saturazione della memoria heap sarà di $11.662233777289416g$.

Capitolo 5

RBD

5.1 Esercizio 1

5.1.1 Traccia

Find the $R(t)$ and MMTF for the system whose reliability diagram is given below.

In calculating MTTF, assume all components are identical and fail randomly with failure rate λ .

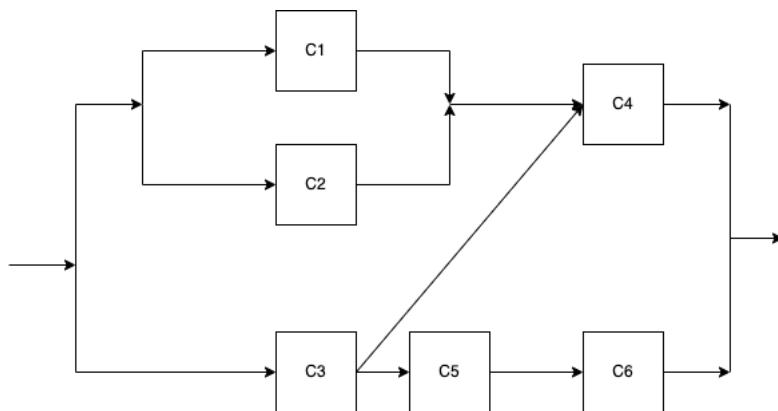


Figura 5.1: Sistema completo

5.1.2 Calcolo della reliability

Innanzitutto, è possibile osservare che il sistema non presenta una struttura serie-parallelo classico. Per calcolarne quindi in maniera più affidabile la reliability è possibile utilizzare la tecnica del **conditioning**, la quale sfrutta il teorema della probabilità condizionata, andando ad effettuare il condizionamento su un componente chiave del diagramma. Dal momento che la reliability $R(t)$ di un sistema è definita come la probabilità che il sistema non fallisca fino all'istante t , è possibile ottenere la reliability tramite la formula:

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

dove

- R_m è la reliability dell' m -esimo componente
- $P(system\ works|m\ works)$ è la probabilità che il sistema funzioni condizionato dal componente m funzioni
- $P(system\ works|m\ fails)$ è la probabilità che il sistema funzioni condizionato dal componente m non funzioni

Osservando il sistema totale è possibile notare la presenza di un serie e di un parallelo, ovvero rispettivamente i componenti C_1, C_2 e i componenti C_5, C_6 . Il sistema equivalente è in figura 5.2.

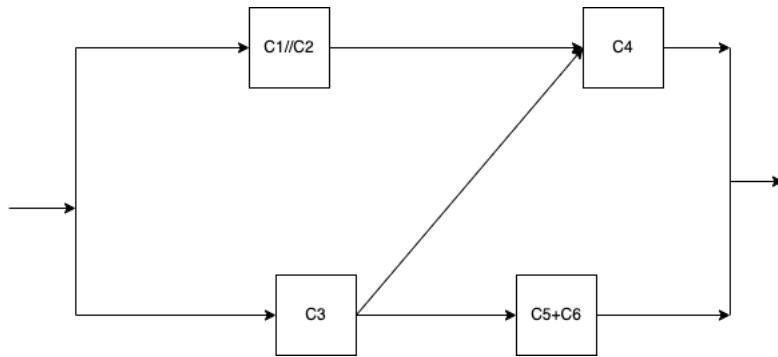


Figura 5.2: Sistema equivalente

Procediamo quindi al calcolo delle reliability:

$$R_{C1//C2} = 1 - [(1 - R_1)(1 - R_2)] = R_1 R_2 - R_1 - R_2 = R_{1/2}$$

$$R_{C5+C6} = R_5 R_6 = R_{56}$$

Condizioniamo ora il sistema al funzionamento del **componente C4**. Calcolando quindi la reliability totale come

$$R_{sys} = R_4 P(system \ works | C4 \ works) + (1 - R_4) P(system \ works | C4 \ fails)$$

C4 fails

Considerando il sistema in figura 5.2 e condizionandolo al fallimento del componente *C4* il sistema risultante è il seguente:

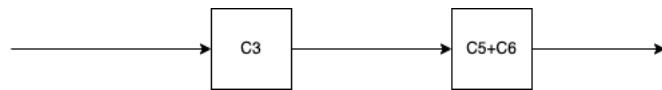


Figura 5.3: Sistema nel caso del fallimento di C4

Possiamo calcolare la probabilità che il sistema funzioni condizionata al fallimento di *C4*

$$P(system \ works | C4 \ fails) = R_3 R_{56}$$

C4 works

Considerando il sistema in figura 5.2 e condizionandolo al funzionamento del componente $C4$ il sistema risultante è il seguente:

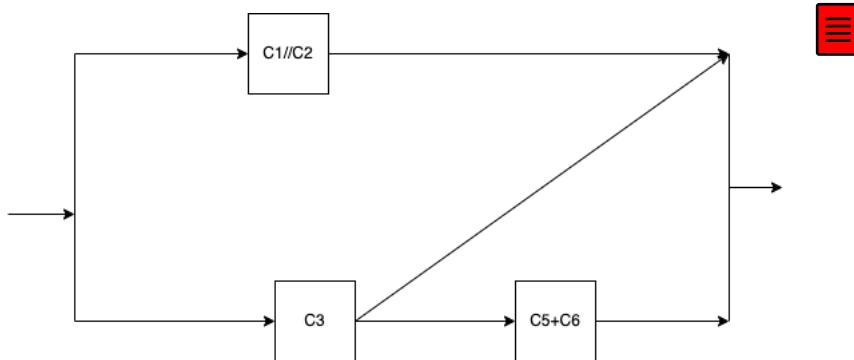


Figura 5.4: Sistema nel caso del funzionamento di $C4$

Possiamo calcolare la probabilità che il sistema funzioni condizionata al funzionamento di $C4$

$$P(\text{system works} | C4 \text{ works}) = 1 - [(1 - R_{1/2})(1 - R_3)]$$

La reliability totale del sistema è quindi:

$$R_{sys} = R_4[1 - [(1 - R_{1/2})(1 - R_3)]] + (1 - R_4)R_3R_{56}$$

5.1.3 Calcolo del MTTF

Considerando che i componenti sono identici e hanno la stessa failure rate λ avremo che $R_m = e^{-\lambda t}$. Possiamo quindi calcolare l'MTTF del sistema utilizzando la reliability totale trovata precedentemente.

$$\begin{aligned} R_{sys} &= e^{-\lambda t}[1 - [(1 - 2e^{-2\lambda t} + e^{-2\lambda t})(1 - e^{-\lambda t})] + (1 - e^{-\lambda t})e^{-3\lambda t} = \dots \\ &\dots = 3e^{-2\lambda t} - 2e^{-3\lambda t} \end{aligned}$$

Sapendo che $MTTF(\lambda) = \int_0^{+\infty} ae^{-b\lambda t} dt = \frac{a}{b\lambda}$ otteniamo

$$MTTF(\lambda) = \int_0^{+\infty} R_{sys}(t) dt = \frac{3}{2\lambda} - \frac{2}{3\lambda} = \frac{5}{6\lambda}$$

5.2 Esercizio 2

5.2.1 Traccia

We want to compare two different schemes of increasing reliability of a system using redundancy. Suppose that the system needs s identical components in series for proper operation. Further suppose that we are given $(m \times s)$ components. Given that the reliability of an individual component is r , derive the expressions for the reliabilities of two configurations. For $m = 5$ and $s = 3$, compare the two expressions as function of a mission time t . Let MTTF of the component be 1400 hours. Out of the two schemes shown in the figure below, which one will provide a higher reliability? Modify the scheme that has lower reliability in order to reach the same reliability of the other given the above MTTF (1400 h).

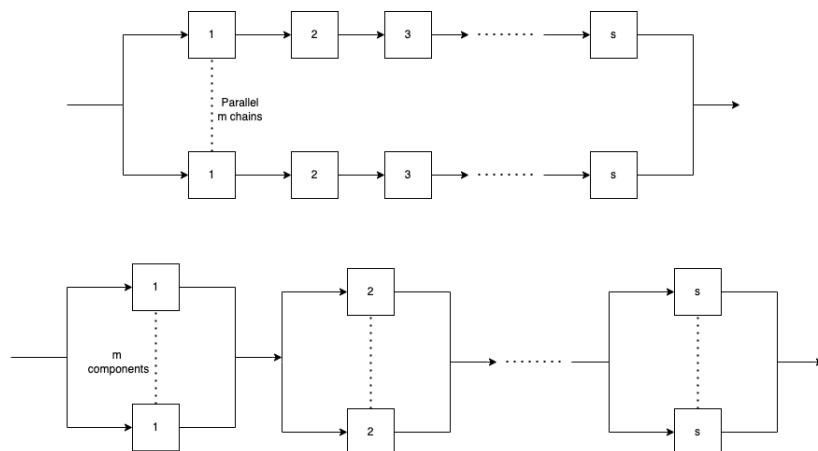


Figura 5.5: Sistemi

5.2.2 Svolgimento

Punto 1

Dal momento che entrambe le configurazioni sono riconducibili ad una struttura serie-parallelo, è possibile calcolare semplicemente la reliability di entrambi:

Sistema 1: il primo sistema si può vedere come un parallelo di m serie, composta ognuna da s componenti. Otteniamo quindi:

$$R_{sys} = 1 - \prod_{i=1}^m (1 - r^s) = 1 - (1 - r^s)^m$$

Sistema 2: il secondo sistema si può vedere come un serie di s paralleli, composta ognuno da m componenti. Otteniamo quindi:

$$R_{sys} = (1 - \prod_{i=1}^m (1 - r))^s = (1 - (1 - r)^m)^s$$

Punto 2

Fissati $m = 5$ e $s = 3$ è possibile confrontare i due sistemi valutando il numero di success path in essi contenuti. In figura 5.6 è riportato il primo sistema, mentre, in figura 5.7 il secondo.

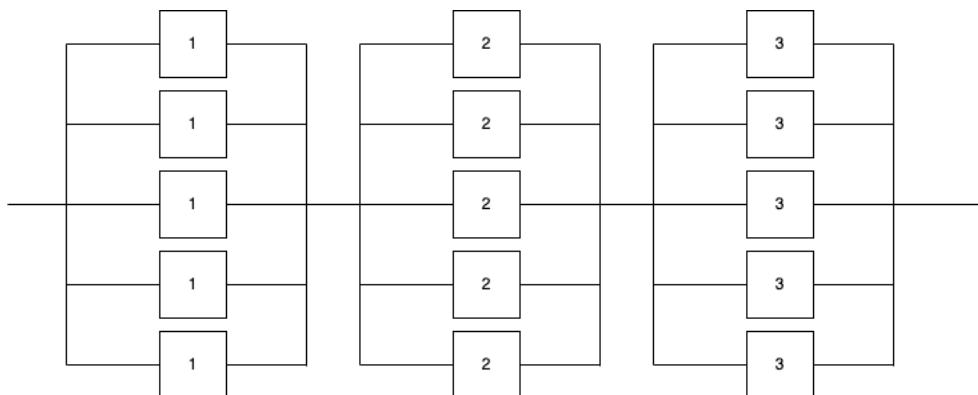
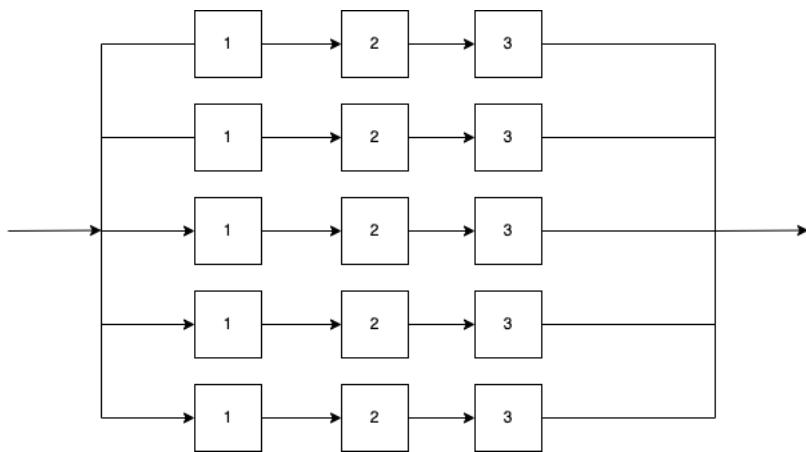


Figura 5.6: Primo sistema per $m=5$ e $s=3$

secondo
sistema

Figura 5.7: ~~Secondo~~ sistema per $m=5$ e $s=3$

primo

Possiamo notare come nel primo sistema siano presenti unicamente m success path, in quanto avendo m serie se uno dei componenti nella serie fallisce allora l'intero success path risulterà inutilizzabile. Nel secondo sistema invece sono presenti m^s success path, in quanto, tale risultato è ottenibile calcolando tutte le possibili combinazioni. Di conseguenza il sistema con maggiore reliability è il secondo, **125 success path** contro i **5 success path** del primo.

Punto 3

Come richiesto dato un MTTF pari a 1400 è stato rappresentato l'andamento della reliability al variare del mission time dei due sistemi, dalla figura 5.8 possiamo notare come il ragionamento prima esplicitato sia corretto.

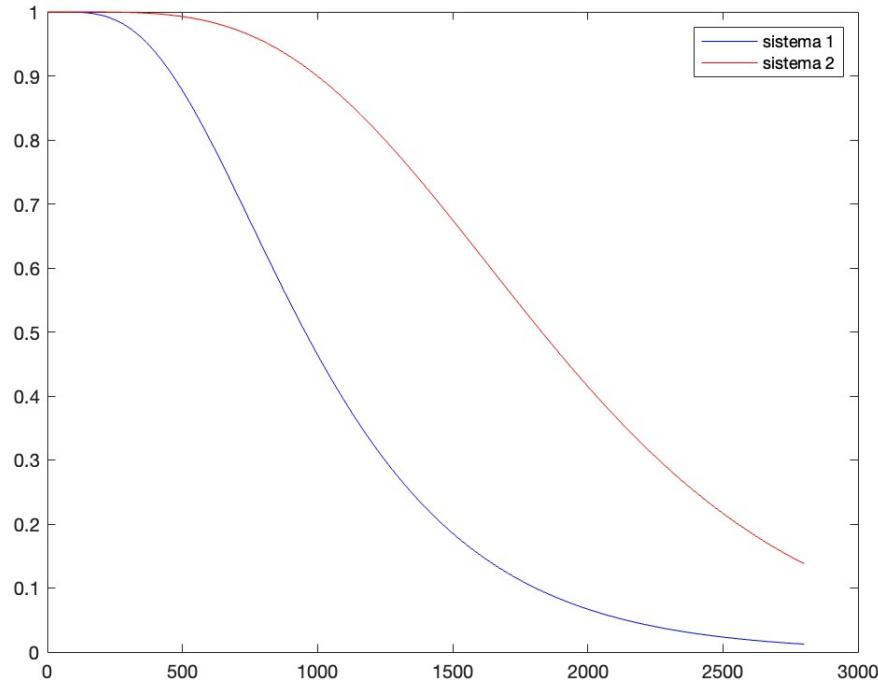


Figura 5.8: Reliability dei sistemi per m=5 e s=3

Punto 4

Per ottenere la reliability uguale per entrambi i sistemi, è necessario aumentare quella del primo sistema. Il valore a cui arrivare dipende dal numero di success paths del secondo sistema: infatti, per poter rendere uguali a livello di reliability i due sistemi è necessario che abbiano lo stesso numero di success paths. Considerando che il numero di success path del primo sistema dipende dal numero di parallelis in esso presenti chiameremo m_1 l'incognita da trovare. Per calcolare il numero di success paths si egualgiano le formule di reliability:

$$R_{sys1} = R_{sys2} \longrightarrow 1 - (1 - r^s)^{m_1} = (1 - (1 - r)^m)^s$$

Siccome il valore di m_1 è quello che indica nel primo sistema il numero di success paths, è necessario fissare gli altri parametri. Posti $m = 5$ e $s = 3$ si ha:

$$1 - (1 - r^3)^{m_1} = (1 - (1 - r)^5)^3 \longrightarrow \log_{1-r^3}(1 - r^3)^{m_1} = \log_{1-r^3}[1 - (1 - (1 - r)^5)^3] =$$

Effettuando un cambio di base, con nuova base e

$$m_1 = \frac{\ln[1 - (1 - (1 - r)^5)^3]}{\ln(1 - r^3)}$$

$$\begin{aligned} r(t) &= \exp(-\lambda t) \\ \lambda &= \exp(-1/MTTF) \end{aligned}$$

Per il calcolo del valore effettivo di m_1 , si è scelto di considerare un mission time di 1100 ore, per andare a definire il valore della reliability dei componenti per cui si avesse tale mission time, avendo un MTTF pari a 1400. Calcolando tale valore si ottiene:

$$r(1100) = \exp^{-\frac{1100}{1400}} = 0.4558$$

Dato tale valore possiamo calcolare m_1 :

$$m_1 = \frac{\ln[1 - (1 - (1 - 0.4558)^5)^3]}{\ln(1 - 0.4558^3)} \simeq 20$$

Il numero di success paths necessari ad avere uguale reliability per il sistema 1 è quindi pari a 20. In figura 5.9 il grafico di confronto tra il sistema modificato e il sistema 2.

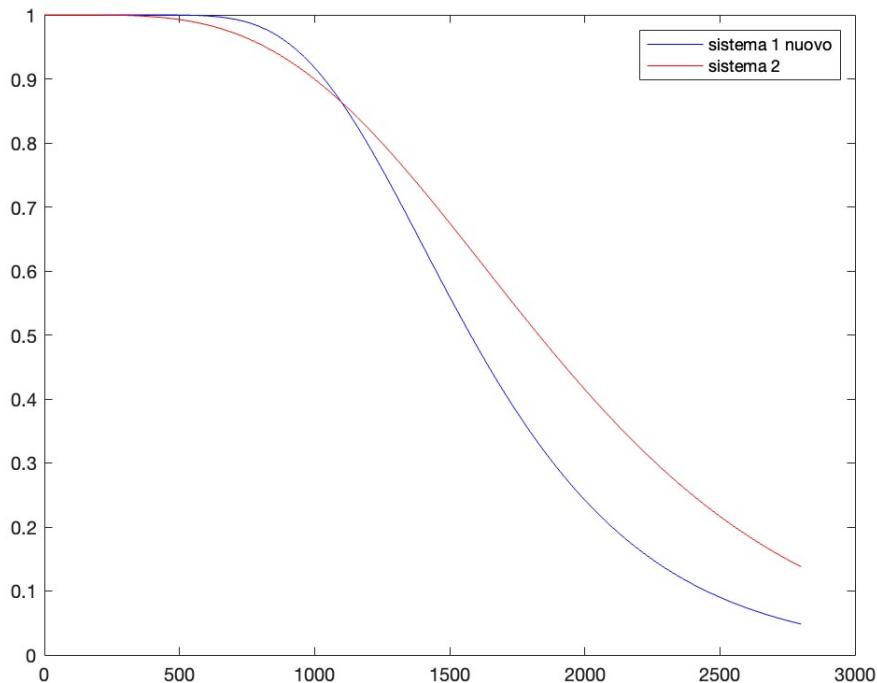


Figura 5.9: Reliability dei sistemi al variare di t

5.3 Esercizio 3

5.3.1 Traccia

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

obeys the exponential failure law and the failure rate of each node is 0.005 failures per hour, determine the reliability of the system at the end of a 48-hour period.

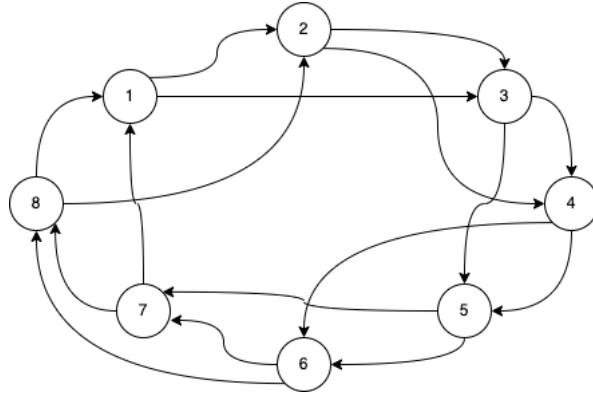


Figura 5.10: Skip-ring network

5.3.2 Svolgimento

Punto 1

Per il calcolo della reliability della rete skip-ring, è necessario andare a identificare tutti i possibili scenari di fallimento. Nel caso in esame, si ha che non possono fallire un numero di nodi superiore a 4. Analizzando caso per caso i vari tipi di fallimento si hanno le seguenti considerazioni:

- Se non si ha il fallimento di alcun nodo varrà la serie tra i nodi, e tale valore sarà indicato come $P(sys|0 \text{ node fails})$
- Se si ha il fallimento di un solo nodo, si avranno 8 differenti serie di componenti, a seconda del nodo che è fallito. Tale valore indica la $P(sys|1 \text{ node fails})$ e sarà quindi moltiplicato per il numero di combinazioni possibili.

coeff bin(8,2) - set di nodi adiacenti, poichè non possono fallire due nodi adiacenti

CAPITOLO 5. RBD

- Se si ha il fallimento di 2 nodi, si avranno 20 combinazioni diverse di serie tra componenti, ognuna delle quali è possibile definirla dalla $P(sys|2 \text{ node fails})$. Per il totale è necessario moltiplicare tale probabilità per il numero di combinazioni, cioè 20

falliscono 3 nodi
adiacenti -> 8
comb
fall 2 adiance e
un terzo non

- Se si ha il fallimento di 3 nodi, si avranno 16 possibili serie di componenti, ognuna delle quali è identificata tramite la $P(sys|3 \text{ node fails})$. Per ottenere la reliability di tutte le configurazioni occorre moltiplicare tale valore per 16
- Se si ha il fallimento di 4 nodi, si hanno solo 2 combinazioni possibili di serie tra i componenti, e ognuna di queste serie è caratterizzata da $P(sys|4 \text{ node fails})$

La reliability totale sarà quindi la somma delle reliability trovate per ogni caso.

Si ottiene:

$$\begin{aligned}
 R_{sys} = P(sys) &= P(sys|0 \text{ node fails}) + 8P(sys|1 \text{ node fails}) + 20P(sys|2 \text{ node fails}) \\
 &\quad + 16P(sys|3 \text{ node fails}) + 2P(sys|4 \text{ node fails}) = \\
 &= R_m^8 + 8R_m^7(1 - R_m) + 20R_m^6(1 - R_m)^2 + 16R_m^5(1 - R_m)^3 + 2R_m^4(1 - R_m)^4
 \end{aligned}$$

Punto 2

Supponendo che la reliability di ogni componente abbia un andamento esponenziale $\exp^{-\lambda t}$, si vuole calcolare la reliability ad un mission time pari a 48. Sostituendo nell'espressione della reliability $\lambda = 0.005$, e valutando l'espressione al punto $t = 48$, si ottiene

$$R_{sys}(48) \simeq 0.729$$

5.4 Esercizio 4

5.4.1 Traccia

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

- MTTFA = 900 h
- MTTFB = 7000 h
- MMTFC = 1000 h

5.4.2 Svolgimento

Per il calcolo delle reliability, essendo ogni configurazione esprimibile come combinazioni di serie e paralleli, è sufficiente applicare le regole di risoluzione per ogni caso. Definiamo le reliability di ogni componente nel seguente modo:

- $R_a(t) = \exp -\frac{t}{900}$



- $R_b(t) = \exp -\frac{t}{7000}$



- $R_c(t) = \exp -\frac{t}{1000}$



Coppia 1

Nel primo caso, il sistema 1 presenta un parallelo di due serie, ognuna composta da 2 componenti, mentre il sistema 2 presenta la serie tra il componente a, e il parallelo dei componenti b e c. Le reliability calcolate hanno quindi tali espressioni:

$$R_{sys1} = 1 - (1 - R_a R_b)(1 - R_a R_b) = \exp^{-0,00211t} + \exp^{-0,00125t} - \exp^{-0,00336t}$$

$$R_{sys2} = R_a(1 - (1 - R_b)(1 - R_c)) = \exp^{-0,00211t} + \exp^{-0,00125t} - \exp^{-0,00225t}$$

Mediante analisi grafica (figura 5.11) è possibile stabilire che il *sistema 1* è più reliable del *sistema 2*.

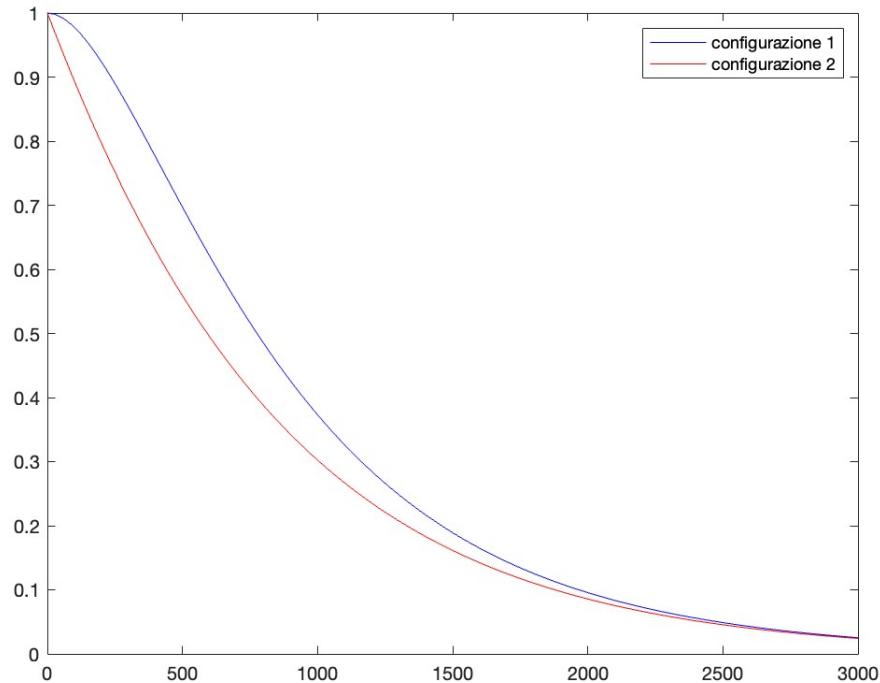


Figura 5.11: Confronto reliability delle configurazioni

Coppia 2

Nel secondo caso, il sistema 1 presenta un serie tra un componente e un parallelo, mentre il sistema 2 è formato da un solo componente. Le reliability calcolate hanno quindi tali espressioni:

$$R_{sys1} = R_a[1 - (1 - R_a)(1 - R_b)] = \exp^{-0,00125t} + \exp^{-0,00222t} - \exp^{-0,00236t}$$

$$R_{sys2} = R_a = \exp^{-0.00111t}$$

Mediante analisi grafica (figura 5.12) è possibile stabilire che il *sistema 2* è più reliable del *sistema 1*.

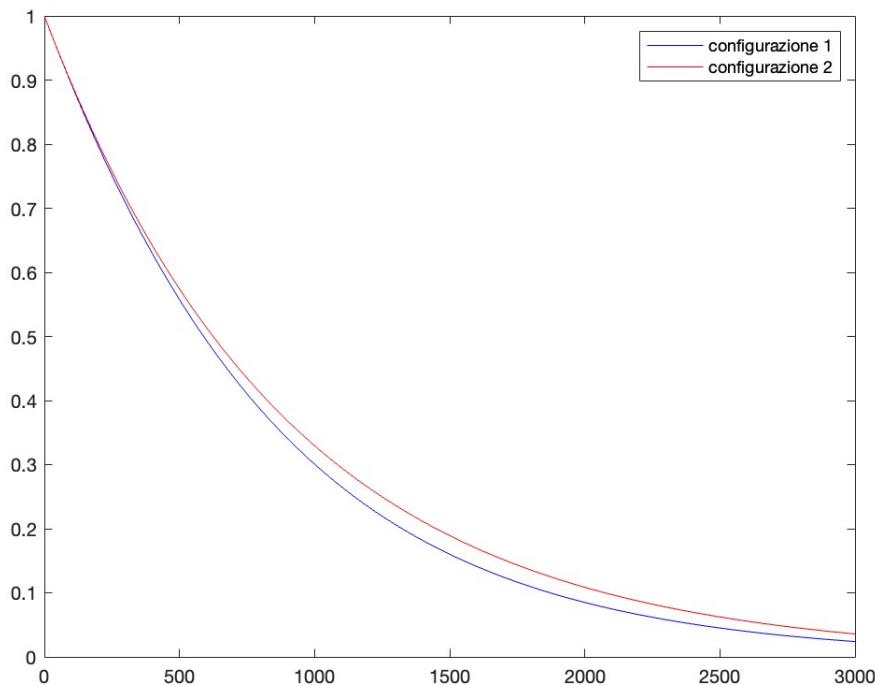


Figura 5.12: Confronto reliability delle configurazioni

Coppia 3

Nel terzo caso, il sistema 1 presenta una serie tra i componenti a e b e un parallelo degli stessi, mentre il sistema 2 è formato dalla sola serie dei due componenti a e b. Le reliability calcolate hanno quindi tali espressioni:

$$R_{sys1} = R_a R_b [1 - (1 - R_a)(1 - R_b)] = \exp^{-0.00139t} + \exp^{-0.00236t} - \exp^{-0.00250t}$$

$$R_{sys2} = R_a R_b = \exp^{-0.00125t}$$

Mediante analisi grafica (figura 5.13) è possibile stabilire che il *sistema 2* è più reliable del *sistema 1*.

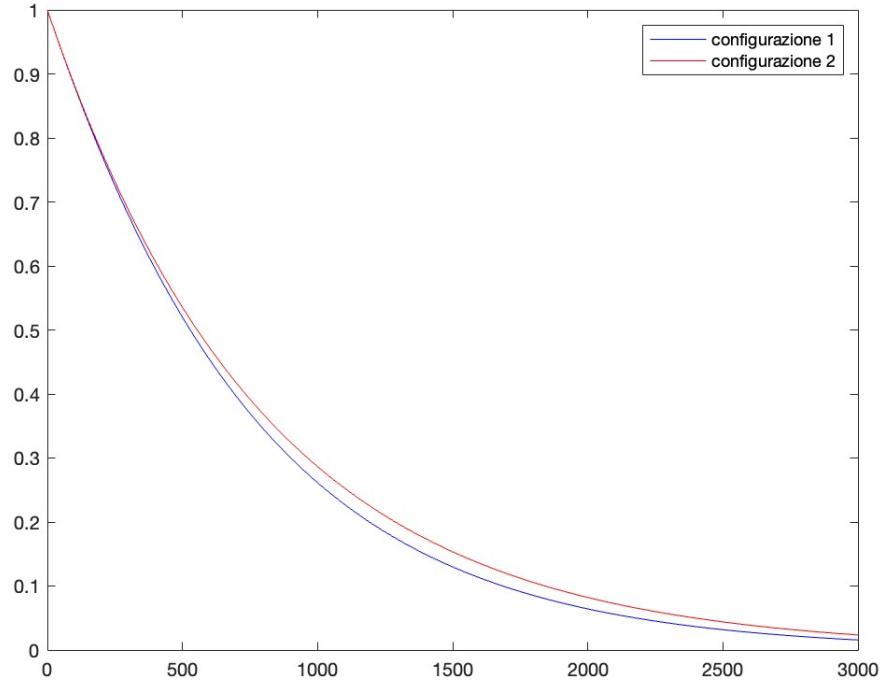


Figura 5.13: Confronto reliability delle configurazioni

Coppia 4

Nel terzo caso, il sistema 1 presenta un parallelo tra il componente a e la serie dei componenti a e b, mentre il sistema 2 è formato dal solo componente a. Le reliability calcolate hanno quindi tali espressioni:

$$R_{sys1} = [1 - (1 - R_a)(1 - R_a R_b)] = \exp^{-0.00111t} + \exp^{-0.00125t} - \exp^{-0.00236t}$$

$$R_{sys2} = R_a = \exp^{-0.00111t}$$

Mediante analisi grafica (figura 5.14) è possibile stabilire che il *sistema 1* è più reliable del *sistema 2*.

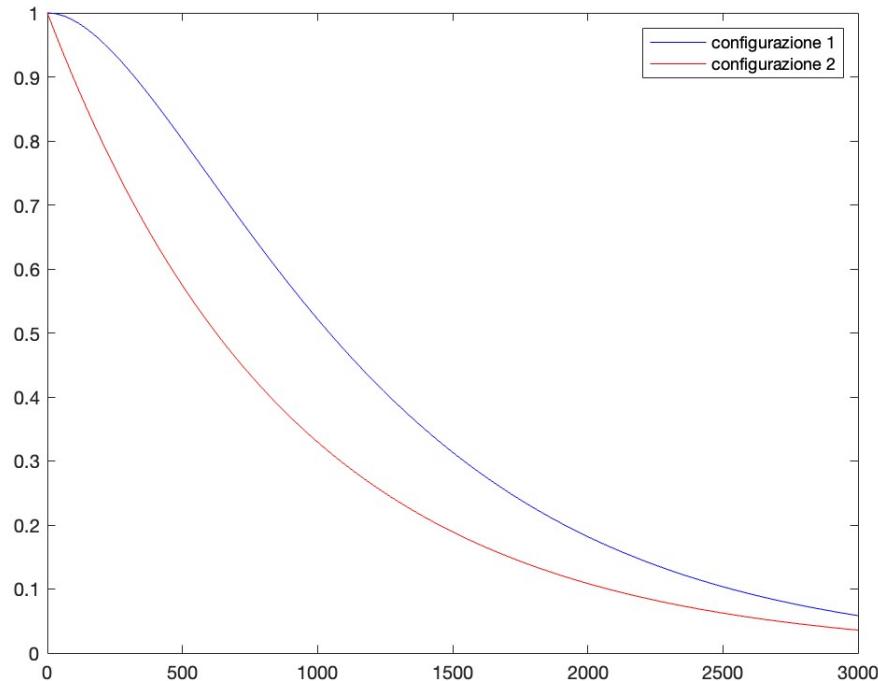


Figura 5.14: Confronto reliability delle configurazioni

5.5 Esercizio 5

5.5.1 Traccia

The system shown in the figure below is a processing system for a helicopter. The system has dual-redundant processors and dual-redundant interface units. Two buses are used in the system, and each bus is also dual-redundant. The interesting part of the system is the navigation equipment. The aircraft can be completely

navigated using the **Inertial Navigation System (INS)**. If the **INS** fails, the aircraft can be navigated using the combination of the Doppler and the altitude heading and reference system (**AHRS**). The system contains three **AHRS** units, of which **only one is needed**. This is an example of functional redundancy where the data from the **AHRS** and the Doppler can be used to replace the **INS**, if the **INS** fails. Because of the other sensors and instrumentation, both buses are required for the system to function properly regardless of which navigation mode is being employed.

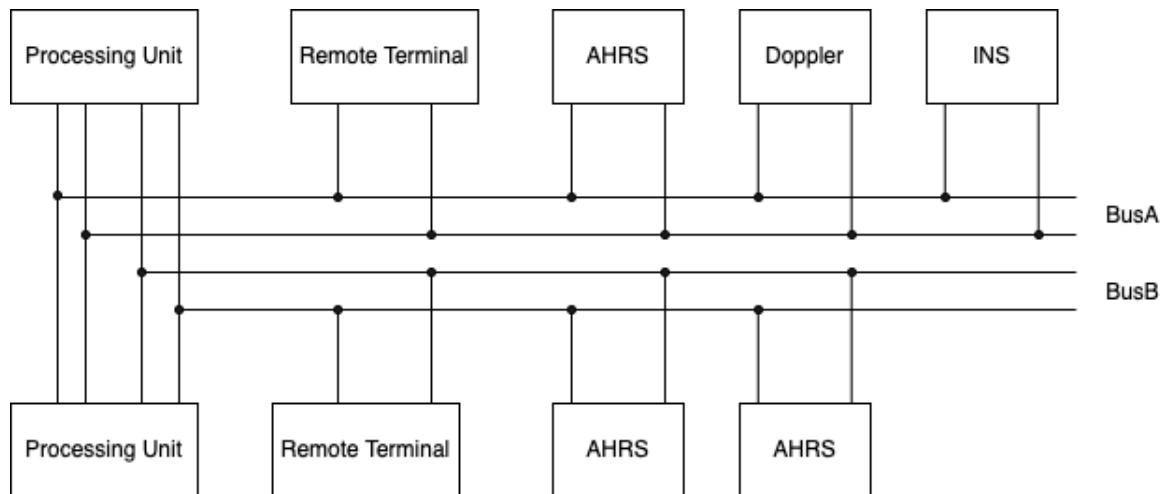


Figura 5.15: Schema RBD

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

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

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

Punto 1

Per tracciare l'RBD di tale sistema, è necessario fare alcune considerazioni sulla composizione del sistema stesso, al fine di delineare come saranno disposti gli elementi all'interno dell'RBD.

Avendo una configurazione dual-redundant per i bus, è necessario avere un parallelo tra ogni coppia di bus A e bus B, che devono essere posti in successivamente in serie, visto che è necessaria almeno una linea di bus di entrambi per il corretto funzionamento del sistema.

Siccome è presente una configurazione dual-redundant anche per i blocchi PU e RT, si avrà, in serie ai bus anche i paralleli delle due copie ridondanti di Processing Unit e Remote Terminal, visto che per il funzionamento del sistema è necessario

almeno un componente PU e un componente RT.

Successivamente, in serie a tutto quello che si è trovato fino ad ora, c'è il blocco relativo alla navigazione. Siccome la navigazione dell'elicottero è garantita se è funzionante il sistema INS, oppure la combinazione di un Doppler e di almeno un componente AHRS, allora l'RBD relativo ai sistemi di navigazione si compone di un parallelo tra il sistema INS e la serie del Doppler e del parallelo dei 3 componenti AHRS. Possiamo quindi procedere alla costruzione del seguente RBD (figura 5.16)

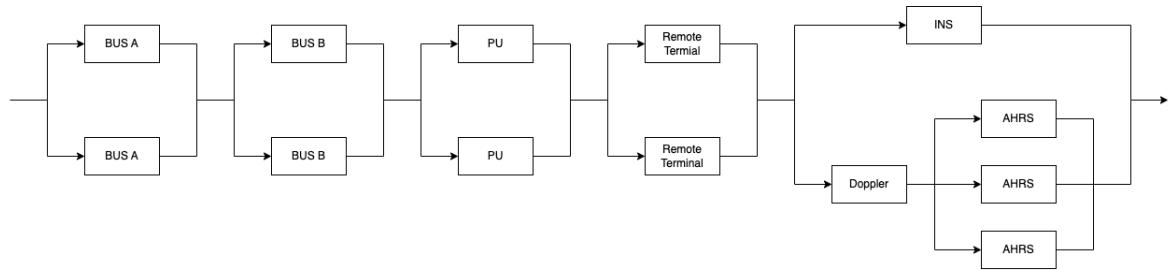


Figura 5.16: Schema RBD

Punto 2

A partire dal diagramma RBD del sistema, la costruzione del relativo Fault Tree risulta alquanto semplice. Infatti a componenti in serie corrispondono componenti in OR all'interno dell'albero, mentre un parallelo tra due componenti si traduce nell'AND di tali componenti all'interno dell'albero. Considerate tali regole, per l'RBD costruito precedentemente si ha il seguente Fault Tree (figura 5.17)

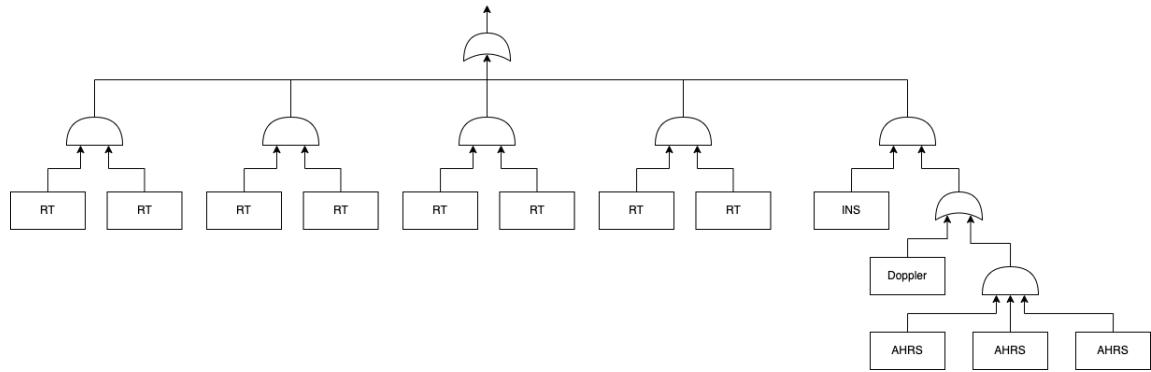


Figura 5.17: Fault Tree

Un cut-set in un FT è un insieme di basic event i quali, se occorrono simultaneamente, fanno sì che occorra anche il top level event, ovvero il fallimento del sistema. Un cut-set è detto minimal cut-set se non può essere ridotto senza perdere il suo status di cut-set. Al fine di evidenziare i minimal cut-set del sistema, è utile esprimere il Fault Tree in forma analitica tramite la sua funzione di struttura, che associa ad ogni AND il prodotto dei componenti che concorrono all'operazione, mentre le OR vengono tradotte come delle somme. Da ciò si ottiene:

$$\begin{aligned}
 \Sigma &= (BUS_A * BUS_B) + (BUS_B * BUS_B) + (PU * PU) + (RT * RT) + [INS * \\
 &\quad (Doppler + (AHRS * AHRS * AHRS))] = \\
 &= (BUS_A * BUS_A) + (BUS_B * BUS_B) + (PU * PU) + (RT * RT) + (INS * \\
 &\quad Doppler) + (AHRS * AHRS * AHRS * INS)
 \end{aligned}$$

Possiamo ricavare dalla formulazione precedenti i minimal cut-set come i minitermini in essa presenti

- BUS_A, BUS_A
- BUS_B, BUS_B

- PU, PU
- RT, RT
- $INS, Doppler$
- $AHRS, AHRS, AHRS, INS$

Punto 3

Per il calcolo della reliability del sistema è possibile affidarsi alle formule di risoluzione degli RBD per serie e parallelo, dal momento che la struttura del diagramma è esprimibile come serie-parallelo. Ipotizzando un tempo al fallimento esponenziale per ogni componente e i MTTF dati, si ricava che:

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

Calcolando il valore della reliability al tempo $t = 1$, si ottiene il valore cercato dopo 1 ora di volo, che è pari a

$$R_{sys}(1) \simeq 0.99999894132$$

Punto 4

Nelle analisi precedenti si è supposto che ogni sistema sia in grado di rilevare perfettamente un proprio fallimento e conseguentemente disabilitarsi. Considerando invece che un componente possa non riconoscere un suo fallimento e il sistema possa non riadattarsi per evitare il fallimento dello stesso, è necessario considerare un fattore di coverage per alcuni componenti che non sia perfetto. Dai dati forniti,

evince che il componente su cui è necessario considerare un **coverage non perfetto** è la **Processing Unit**. Chiamato c il fattore di coverage, è necessario quindi modificare la formula della reliability nel caso delle PU. Infatti, la **reliability del parallelo dei componenti PU** è data dalla **probabilità che una replica funzioni più la probabilità che tale replica fallisca per la probabilità di rilevare il fallimento, moltiplicato la probabilità che l'altra replica funzioni.**



$$R_{PU} + c(1 - R_{PU})R_{PU}$$

L'espressione della reliability assume quindi la seguente forma:

$$R_{sys} = [1 - (1 - R_{bus})^2][1 - (1 - R_{bus})^2][R_{PU} + c(1 - R_{PU})R_{PU}][1 - (1 - R_{RT})^2][1 - (1 - R_{INS})(1 - R_{Doppler})][1 - (1 - R_{INS})(1 - R_{AHRs})^3]$$

Per ottenere una reliability maggiore di 0.99999 dopo un ora, basta calcolare l'espressione precedente per $t = 1$ e calcolare la disequazione $R_{sys} \geq 0.99999$ nella sola variabile c . Si ottiene:

$$R_{sys} \geq 0.99999 \longrightarrow c \geq 0.91057$$

Capitolo 6

Field Failure Data Analysis

6.1 Traccia

Condurre una FFDA di tipo log-based su due file di log appartenenti ai due seguenti supercalcolatori:

- **Mercury**: sistema del National Center for Supercomputing Application (NCSA) alla University of Illinois
- **Blue Gene/L**: sistema del Lawrence Livermore National Labs (LLNL)

Rispondere poi ai seguenti quesiti:

- Si possono usare le stesse finestre di coalescenza in diversi nodi (sia per Mercury che per BG/L) e categorie di errore (solo per Mercury)?
- Cosa si può dire della relazione tra l'affidabilità del sistema e l'affidabilità dei nodi?

- Esistono colli di bottiglia dell'affidabilità (cioè, i principali contributori al numero totale di fallimenti)?
- Nodi funzionali simili (per esempio, due nodi di calcolo Mercury tg-cX o nodi BG/L I/O Ri:Mx:Nz) presentano parametri di affidabilità simili?
- Esiste una relazione tra i tipi di errore e il nodo (solo per Mercury)?

6.2 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.

In altre parole, consiste nell'analisi dei dati relativi ai fallimenti di un sistema, dove i dati sono raccolti sul campo. Tipicamente tale tecnica prevede il susseguirsi delle seguenti fasi:

- **Data Collection:** si raccolgono i dati del funzionamento del sistema in file tipicamente detti di log. Tali file sono grezzi
- **Data Filtering:** si effettua un filtraggio dei dati raccolti per concentrare l'analisi solo sui dati significativi
- **Data Manipulation:** i dati vengono manipolati al fine di ottenere una corrispondenza tra dati raccolti e fallimenti del sistema
- **Data Analysis:** i dati sono sottoposti ad analisi, per differenti scopi, che possono essere quello di trovare la curva di reliability del sistema, fino all'identificazione della distribuzione del Time to Failure

L'analisi proposta in seguito sui due sistemi precedentemente citati, parte a valle della fase di filtering, dal momento che i file proposti risultano già pre-elaborati.

6.3 Mercury

6.3.1 Descrizione del sistema

Mercury è un super calcolatore composto da diversi componenti, avente un architettura a **3 livelli con un nodo centrale che governa la struttura chiamato master**. Ogni livello ha differenti responsabilità: **login si occupa degli accessi, computation si occupa delle operazioni elaborate, mentre storage gestisce gli accessi alle memorie**. L'architettura è schematizzata nella figura seguente.

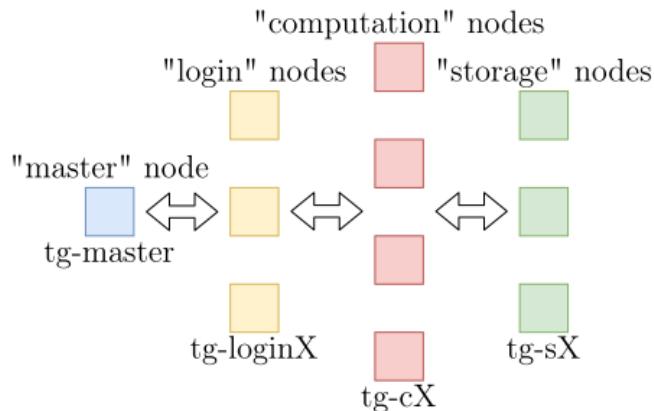


Figura 6.1: Architettura Mercury

I dati da analizzare sono stati raccolti sfruttando un processo demone syslog. A valle dell'esecuzione di tale processo si ottiene il file di log chiamato MercuryErrorLog.txt. Tale file contiene **80854 entry**, che riguardano **solo errori fatali del sistema, formattate come segue**:

- **timestamp:** valore di intero temporale in formato UNIX

- **nodo origine**: stringa del tipo tg-x che identifica la tipologia del nodo in cui si è verificato l'errore
- **sottosistema di origine**: stringa che identifica il sottosistema che ha originato l'errore. Esistono 5 tipi di sottosistemi e un nodo può contenerne più di uno: DEV, MEM, I-O, NET, e PRO
- **messaggio**: stringa che contiene il messaggio dell'errore

6.3.2 Fase di Data Manipulation del sistema

L'obiettivo della fase di manipulation è identificare i fallimenti del sistema. A partire dal log filtrato infatti, si vuole estrarre una relazione tra errori e fallimenti del sistema, ma tale relazione non è mai 1:1.

Per identificare i fallimenti si utilizza ad esempio la tecnica di **coalescenza**, in grado di rilevare la correlazione tra le entry del file di log.

Coalescenza

Un tipo di coalescenza che è possibile utilizzare è quella **temporale**: si parla di coalescenza temporale quando entry differenti vengono raggruppate secondo un algoritmo che misura la distanza tra esse a livello temporale(timestamp). L'algoritmo è detto *tupling*, e i gruppi di entry che si generano vengono dette **tuple**. L'algoritmo lavora come segue: se la differenza tra due timestamp di due entry consecutive è minore di una soglia W, detta **finestra di coalescenza**, allora le due entry fanno parte della stessa tupla.

La scelta cruciale è quindi quella del valore della finestra W. Tale valore influenza i risultati dell'algoritmo:

- se si considera una finestra di coalescenza troppo grande si incorre in **collisioni** tra le entry di una stessa tupla: **guasti differenti saranno contenute in una stessa tupla**
- se viceversa, la finestra di **coalescenza è troppo piccola**, si incorre in **troncamenti**: eventi relativi allo stesso guasto sono inseriti in tuple differenti

Dal punto di vista della reliability empirica del sistema, il verificarsi di collisioni è più critico rispetto al verificarsi di troncamenti. Se si hanno collisioni la stima effettuata sarà infatti ottimistica, mentre se si hanno troncamenti la stima sarà pessimistica.

Sensitivity Analysis

Analizzando la tuple, si tende a preferire una finestra di coalescenza piccola, in modo da effettuare una stima pessimistica della curva di reliability stessa. Si accettano quindi più volentieri troncamenti tra le tuple piuttosto che collisioni. La scelta del valore di W è demandata alla fase di **sensitivity analysis**. Per prima cosa si applica l'algoritmo di tupling al variare di W e, dopo aver graficato l'andamento delle tuple, si sceglie come valore ottimo della finestra di coalescenza, il **valore di W immediatamente successivo al valore corrispondente al gomito** della curva. Nel caso dell'analisi del sistema Mercury, tale fase ha prodotto il seguente andamento

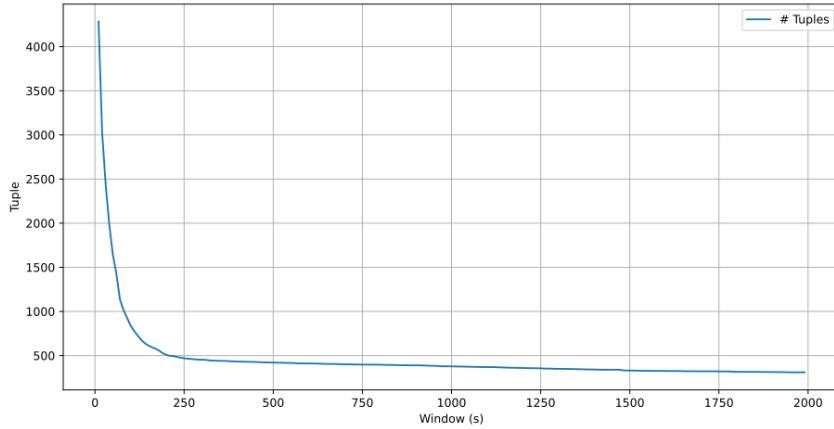


Figura 6.2: Mercury System Sensitivity

Si è quindi deciso di scegliere un valore di finestra pari a **200 secondi** (circa 3 minuti e 20").

Tupling

Una volta fissato il valore della finestra di coalescenza, si passa a raggruppare le entry sfruttando l'algoritmo di tupling con il valore specifico di W , ottenendo un file diverso per ogni gruppo di entry. Si ottengono **508 tuple**. Date le tuple così composte è possibile ricavare alcune informazioni aggiuntive:

- **entries**: numero di entry del log appartenenti alla tupla
- **length**: differenza tra i timestamp della prima e dell'ultima entry della tupla, indica la durata della tupla
- **starting points**: timestamp della prima entry della tupla con annessa durata della stessa

- interarrival: tempo (in secondi) che intercorre tra la prima entry della tupla corrente e l'ultima entry della tupla precedente

Bottleneck

Una prima analisi per verificare la presenza o meno di un bottleneck del sistema è verificare se ci siano nodi particolarmente avvezzi ad errori e se tali errori sono contenuti in un intervallo temporale specifico. Tale analisi è possibile farla andando a considerare il numero di istanze che sono presenti nelle varie tuple; le tuple che conterranno maggiormente errori identificheranno un problema specifico, che se si dimostra essere dovuto ad un singolo componente, è possibile identificare come collo di bottiglia del sistema stesso.

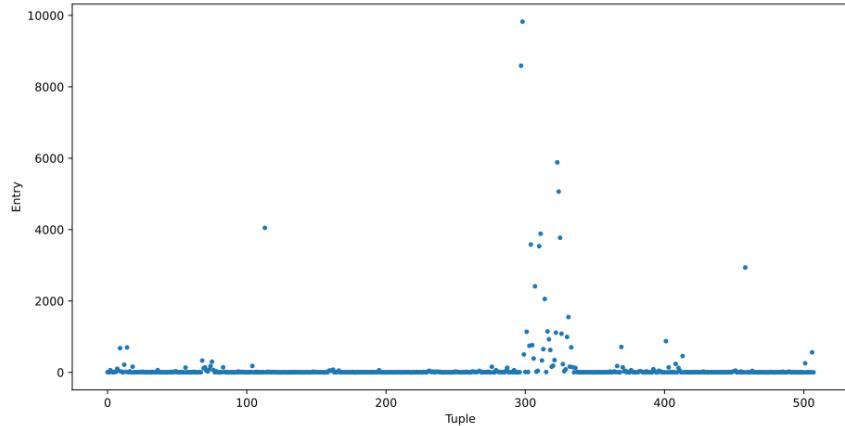


Figura 6.3: Mercury System Scatter Tuple

Trovate quindi le tuple che contengono il maggior numero di entry, si passa allo studio dell'intorno della tupla, per verificare se vi sono interessantemente di un singolo componente che si riterrebbe quindi essere il bottleneck del sistema. Per fare ciò, come evidenziato dal grafico, si nota che la maggioranza delle entry è contenuta

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

tra la tupla 297 e la tupla 330. In questo intervallo sono comprese 60151 istanze, che compongono circa il 74.5% di tutto il file di log. È evidente quindi che in questo intervallo temporale sono concentrati la maggioranza degli errori.

tg-c401	60105
tg-s176	13
tg-c027	9
tg-c106	7
tg-c407	4
tg-master	4
tg-c735	3
tg-c128	2
tg-s044	2
tg-c685	1
tg-c894	1

prendiamo tutte le entry contenute nelle tuple, merge-> groupby nodo

Analizzando nel dettaglio tali tuple, si vede come la maggioranza degli errori è dovuta al nodo computazionale numero 401 (tg-c401), tale esecuzione fa emergere che il nodo tg-c401 è responsabile di circa il 99% degli errori nel file nell'intervallo di tuple 297-330. Inoltre il trend si verifica anche nell'intero file di log, infatti lo stesso nodo è responsabile di circa 77.1% degli errori su tutto il file di log, ed è quindi plausibile che il nodo tg-c401 sia un probabile bottleneck per il sistema. In figura la distribuzione degli errori per nodo su tutto il file di log.

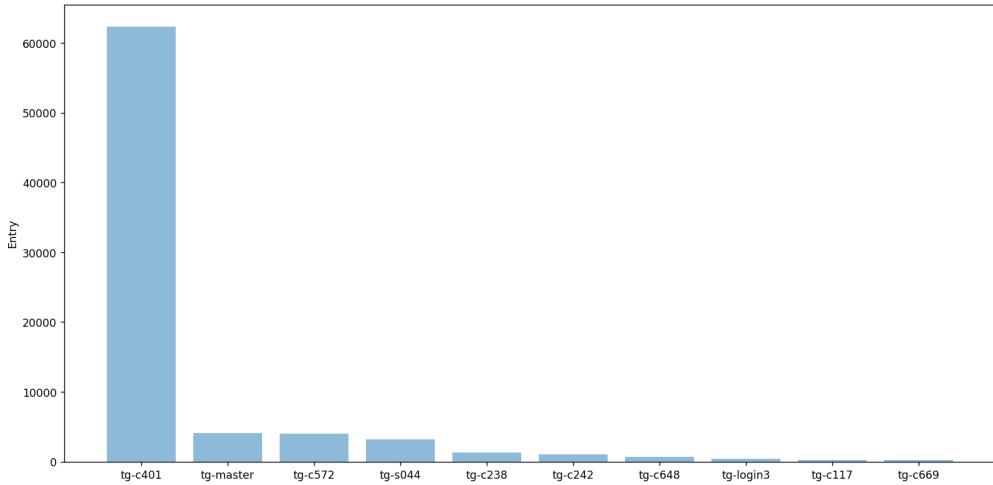


Figura 6.4: Mercury Node Error Count

Analisi Troncamenti

La scelta della finestra di coalescenza influisce sul numero di troncamenti e collisioni commessi a valle del tupling. Si vuole quindi effettuare una stima dei troncamenti che commessi. Per prima cosa si calcola il valore q corrispondente al 10° percentile (0.1 quantile) dei tempi di interarrivo delle tuple. Tale valore è circa 280 secondi(circa 5 minuti), si individuano poi tutte le coppie di tuple adiacenti che distano meno di questo valore. Per ognuna di queste coppie è possibile stimare un troncamento, da questa prima stima si ottengono 51 troncamenti, il che significa che per circa il 10% delle tuple totali potrebbe essersi verificato un troncamento.

51 è il 10% di
508

Analisi Collisioni

Analogamente si vuole effettuare una stima delle collisioni commesse a causa del tupling. Il numero di potenziali collisioni è ottenuto individuando il numero di tuple per le quali si hanno più nodi di origine. Sono quindi stimate 47 collisioni,

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

il che significa che per circa l'9% delle tuple totali potrebbero essersi verificate collisioni. Come è possibile dedurre dalla scarsa percentuale di collisioni, il valore scelto di finestra è da ritenersi valido

Considerazioni collisioni

22284	1171344374	tg-c401	DEV	+BEGIN HARDWARE ERROR STATE AT CPE	computation	298
22285	1171344374	tg-c401	DEV	+END HARDWARE ERROR STATE AT CPE	computation	298
22286	1171344374	tg-c401	DEV	+Platform PCI Component Error Info Section	computation	298
22287	1171344374	tg-c401	DEV	+Platform Specific Error Info Section	computation	298
22288	1171344374	tg-c401	DEV	Component Info: Vendor Id =xx, Device Id =xx, Class Code =xx, Seg Bus Dev Func =xxxxxx	computation	298
22289	1171344374	tg-c401	DEV	Component Info: Vendor Id =xx, Device Id =xx, Class Code =xx, Seg Bus Dev Func =xxxxxx	computation	298
22290	1171344374	tg-c401	DEV	Component Info: Vendor Id =xx, Device Id =xx, Class Code =xx, Seg Bus Dev Func =xxxxxx	computation	298
22291	1171344374	tg-c401	DEV	Component Info: Vendor Id =xx, Device Id =xx, Class Code =xx, Seg Bus Dev Func =xxxxxx	computation	298
22292	1171344374	tg-c401	DEV	+ Platform Specific Error Detail:	computation	298
22293	1171344374	tg-c401	MEM	Physical Address x, Address Mask: x, Node: x, Card: x, Module: x, Bank: x, Device: x, Row: x, Col: x	computation	298
22294	1171344374	tg-c401	MEM	Physical Address x, Address Mask: x, Node: x, Card: x, Module: x, Bank: x, Device: x, Row: x, Col: x	computation	298
22295	1171344376	tg-s176	DEV	+BEGIN HARDWARE ERROR STATE AT CPE	storage	298
22296	1171344376	tg-s176	DEV	+Platform PCI Component Error Info Section	storage	298
22297	1171344376	tg-s176	DEV	Component Info: Vendor Id =xx, Device Id =xx, Class Code =xx, Seg Bus Dev Func =xxxxxx	storage	298

Figura 6.5: Errori

Analizzando i file tuple è possibile fare ulteriori considerazioni sulle possibili propagazioni di fallimenti nel sistema. E' stato analizzato il file tuple-298 poichè è il file che presenta più entries. Analizzando la tupla si osserva che il nodo tgc-401 ha restituito un errore dovuto al malfunzionamento della CPE e PCI. In particolare è presente un errore dovuto al fallito accesso in memoria condivisa (nodi storage), è stato coinvolto anche il nodo di storage tg-s176 che restituisce lo stesso errore del nodo tgc-401. Si può dedurre che si tratta di una propagazione del fallimento delle due componenti PCI e CPE.

6.3.3 Fase di Data Analysis del sistema

Terminata la fase di data manipulation è possibile proseguire con l'analisi dei dati vera e propria. L'obiettivo di questa fase è quello di estrarre dai dati alcune

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

informazioni relative alla reliability del sistema. Per prima cosa è possibile estrarre dai dati la distribuzione di probabilità empirica del time to failure(CDF), ovvero del tempo oltre il quale si verifica un fallimento del sistema. A tal fine, si applicano ai dati relativi ai tempi interarrivo una serie di test di Bontà di adattamento (GoF),

test statistici che verificano la presenza di una distribuzione nota nei dati.

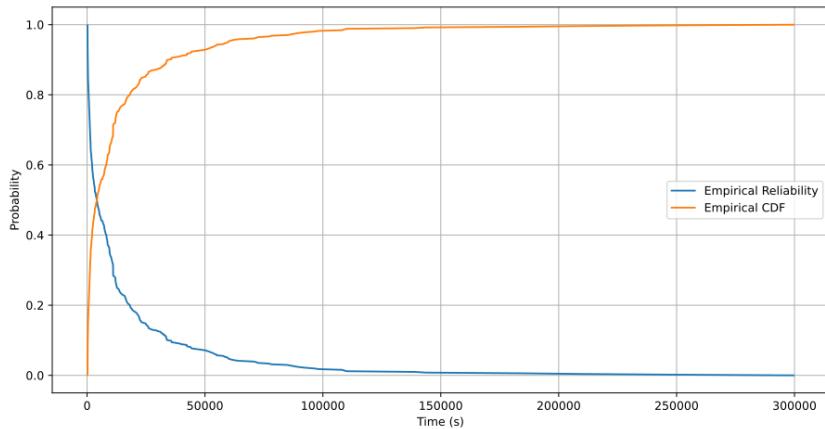


Figura 6.6: Mercury Empirical Reliability

Da tale distribuzione empirica è possibile, attraverso lo strumento di Curve Fitting, testare la bontà di adattamento della curva a varie distribuzioni. Quella evidenziata come curva di fitting migliore è un esponenziale di secondo grado, avente i coefficienti della tabella a lato. Per evidenziare la bontà di tale approssimazione, in prima analisi è possibile verificare il valore di alcuni parametri:

- SSE: rappresenta la Sum of Square Errors, somma quadratica degli errori, e per avere una buona approssimazione deve essere prossimo a zero. Nel caso della GoF eseguita tale valore è pari a 0.1457

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

- R-square: misura il legame tra la variabilità dei dati e la correttezza del modello statistico utilizzato, ed è evidente che deve essere il più possibile simile ad 1. Nel caso di tale modello l'R-quadro è pari a 0.9961

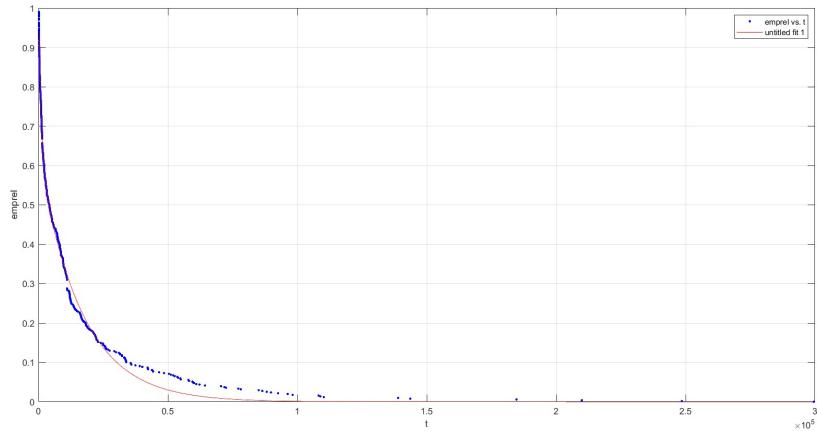


Figura 6.7: Fitting Exp

$$\text{fit-Mercury}(x) = ae^{bx} + ce^{dx}$$

a	b	c	d
0.3922	-0.001036	0.6297	-6.077e-05

E' possibile eseguire il test non parametrico di Kolmogorov-Smirnov: usando tale test è possibile affermare con un livello di significatività di 0.05 che è possibile accettare o rigettare l'ipotesi che i dati provengono dalla distribuzione scelta, avendo i parametri riportati in precedenza. Dai risultati ottenuti è possibile accettare l'ipotesi nulla, confermando la bontà dell'adattamento della reliability empirica ad una reliability esponenziale.

H-value	p-value	k-value
0	0.6666	0.0472

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

È possibile trarre alcune conclusioni da tale test: un tipo di TTF empirica esponenziale è tipicamente associato ad un guasto a livello hardware o firmware, tipicamente semplice, mentre altre tipologie di curve empiriche possono essere associate a degenerazione di un componente, o a guasti di tipo software, o a guasti di natura più complessa. Avendo evidenziato un tipo di curva empirica esponenziale, è facile intuire che probabilmente la reliability del sistema Mercury dipende principalmente da errori e fallimenti che sono riconducibili all'hardware. Tale teoria viene confermata dal fatto che la maggioranza delle entry presenti nel file di log è appartenente alla categoria DEV, che rappresentano errori legati a schede PCI e alle periferiche in generale.

6.3.4 Fase di Data Manipulation dei componenti

Per studiare il comportamento dei vari componenti del sistema Mercury, si è scelto di studiare la reliability in funzione del componente che ha generato il fallimento. Si ripetono le stesse operazioni eseguite in precedenza sull'intero sistema però andando a considerare i dati relativi ad ogni singolo sottosistema. Si vanno quindi a suddividere le diverse entry del log a seconda del sottosistema di origine.

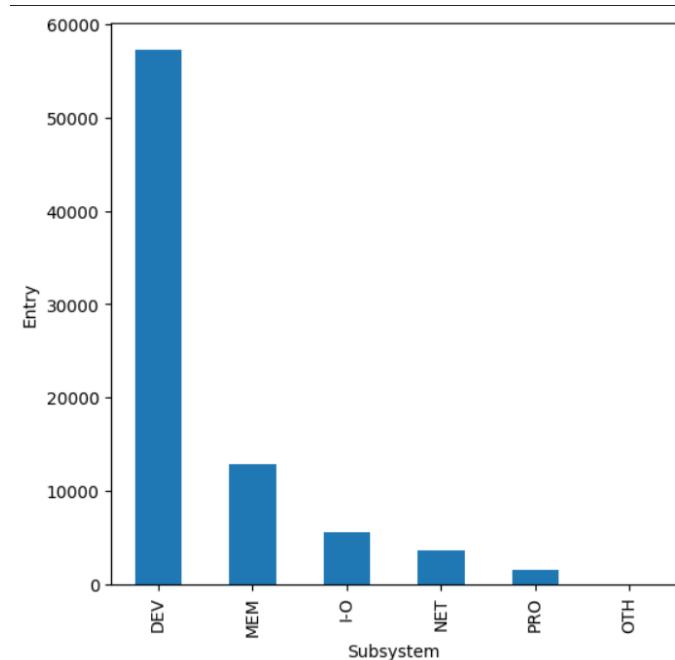


Figura 6.8: Numero di entry per sottosistema

Per analizzare la reliability dei componenti di ogni tipo di sottosistema, si è studiato l'**andamento dell'algoritmo di tupling al variare del valore della finestra di coalescenza**, facendo quindi una sensitivity analysis dei file di log raggruppati per categoria.

Sensitivity Analysis

Si sono ottenute le seguenti curve

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

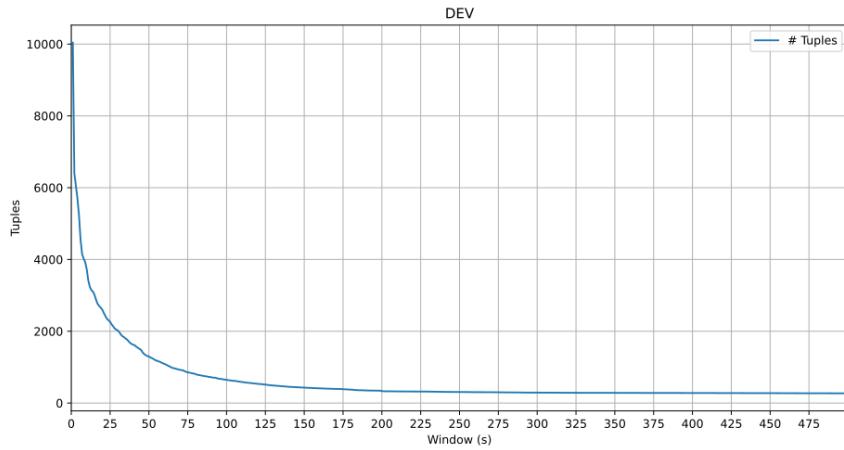


Figura 6.9: Mercury DEV Sensitivity

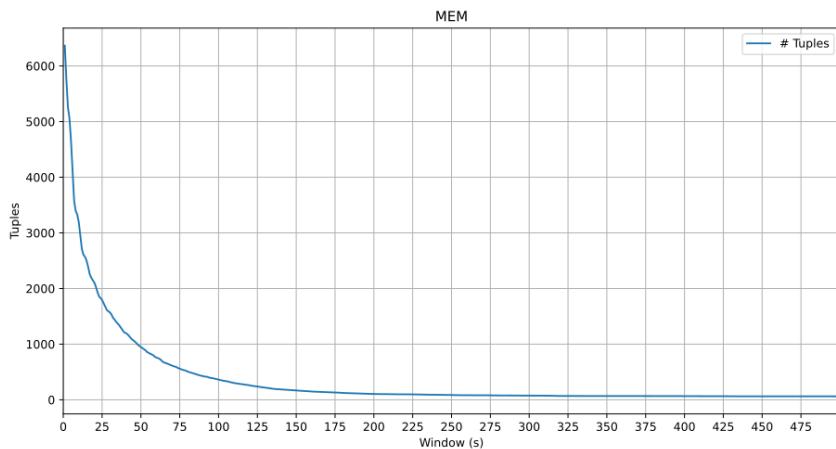


Figura 6.10: Mercury MEM Sensitivity

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

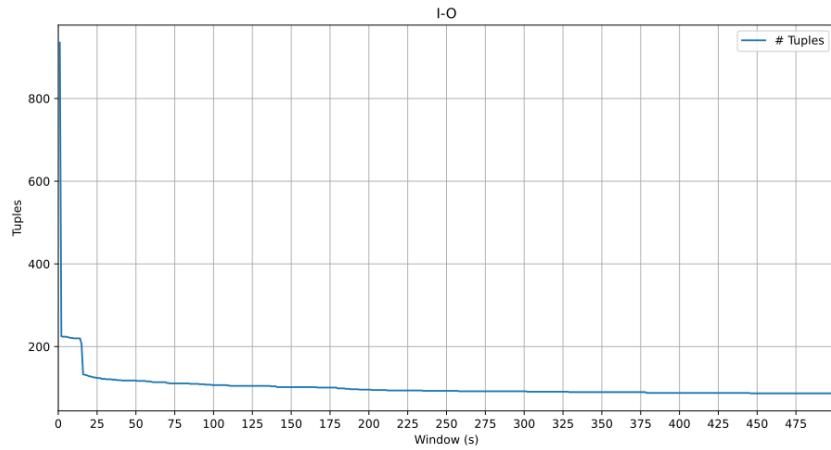


Figura 6.11: Mercury IO Sensitivity

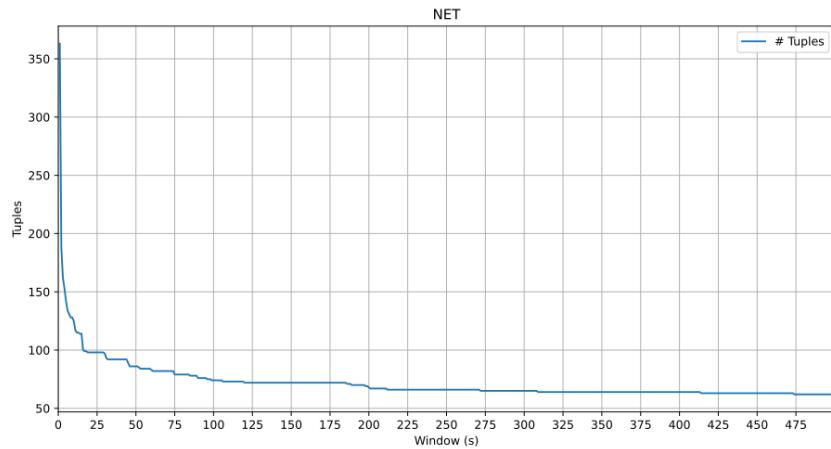


Figura 6.12: Mercury NET Sensitivity

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

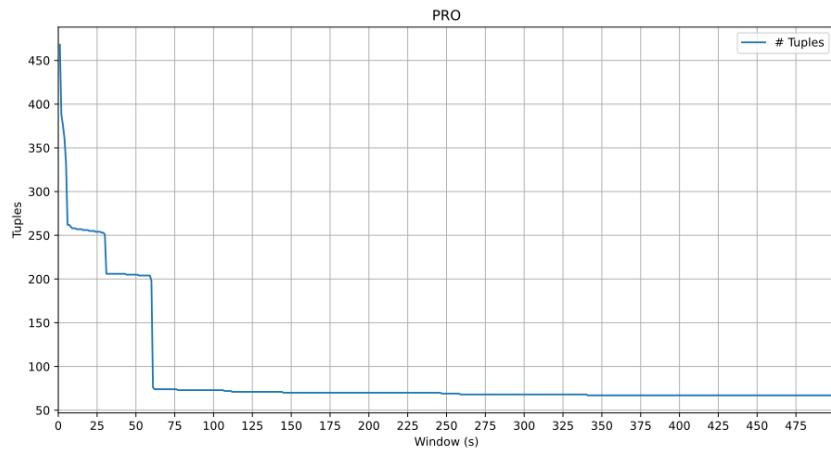


Figura 6.13: Mercury PRO Sensitivity

A valle della sensitivity analysis, per ognuno dei sottosistemi si scelgono le finestre di coalescenza riportate in tabella

Sottosistema	Finestra coalescenza
DEV	200
MEM	200
I-O	100
NET	100
PRO	100

Tupling

Si riporta in tabella il numero di tuple ottenuto per ogni sottosistema eseguendo l'algoritmo di tupling con le finestre di coalescenza riportate in precedenza

Sottosistema	Numero Tuple
DEV	334
MEM	105
I-O	107
NET	74
PRO	73

6.3.5 Fase di Data Analysis dei componenti

Si sono studiate le reliability dei singoli componenti, graficando l'andamento degli interarrivi, esattamente come fatto in precedenza per l'analisi del sistema completo. Sono stati quindi ottenuti i risultati in figura: come era facile pronosticare, il sottosistema DEV è quello che espone la peggiore reliability, vista la gran mole di entry presenti nel log dovuti a tale sottosistema.

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

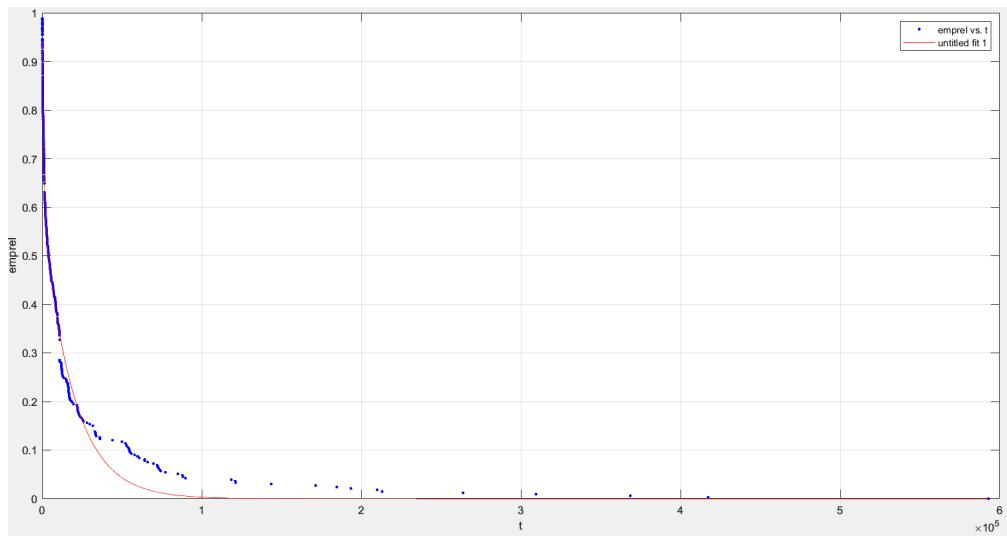


Figura 6.14: Mercury DEV

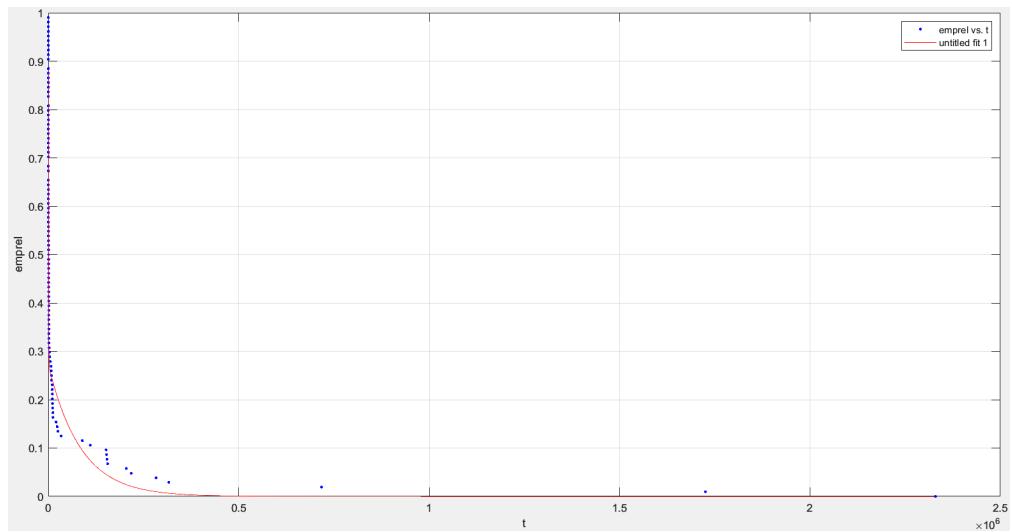


Figura 6.15: Mercury MEM

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

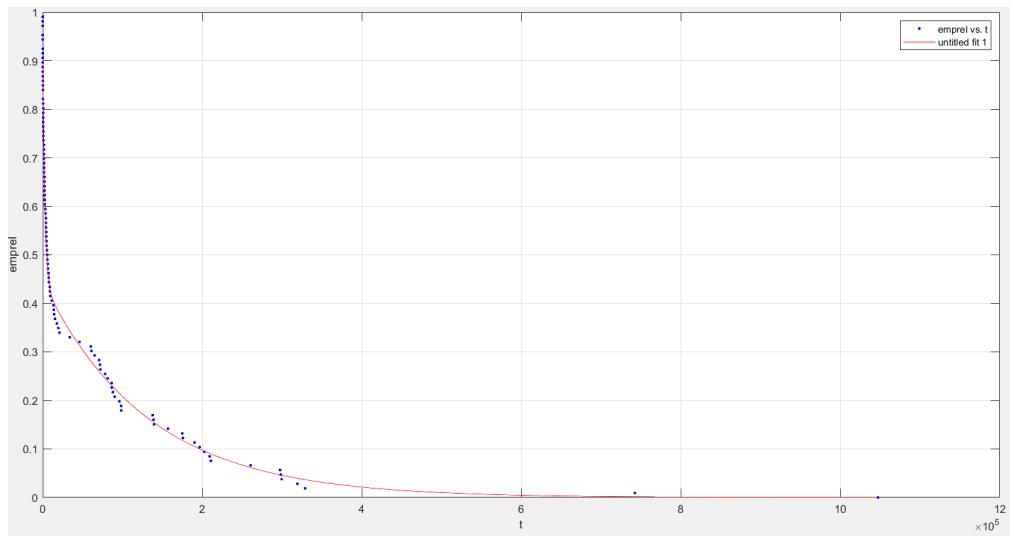


Figura 6.16: Mercury IO

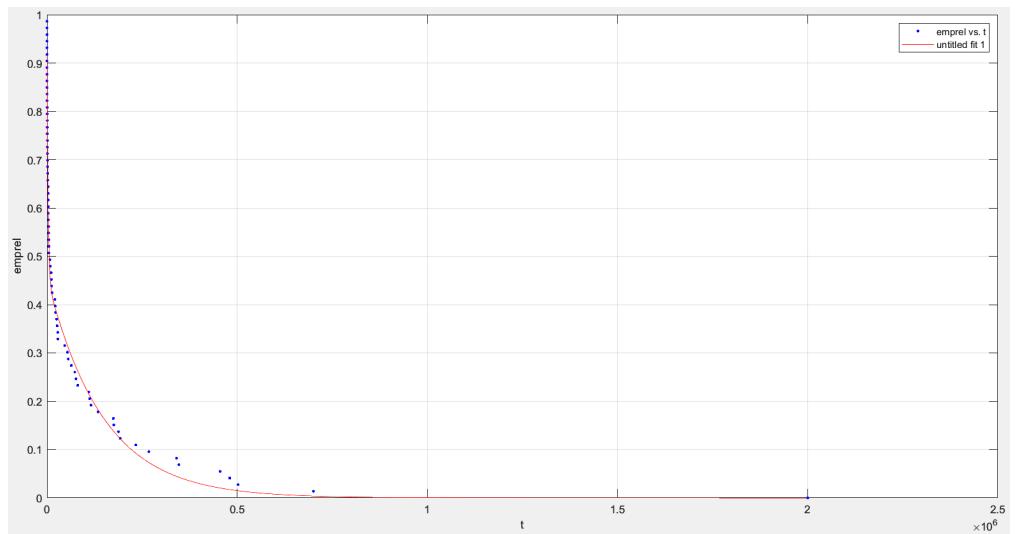


Figura 6.17: Mercury NET

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

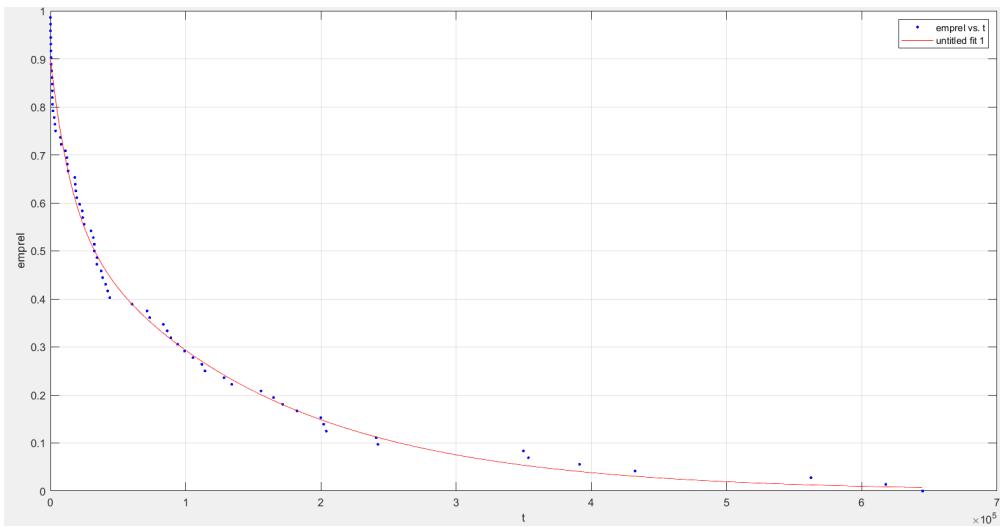


Figura 6.18: Mercury PRO

	Rel teorica	H-val	p-val	k-val
DEV	Exp2	0	0.1986	0.0870
MEM	Exp2	0	0.5560	0.1100
I-O	Exp2	0	0.9659	0.0680
NET	Exp2	0	0.9939	0.0685
PRO	Exp2	0	0.8685	0.0972

A valle dell'ottenimento di tali distribuzioni teoriche, si vanno ad effettuare i test per la verifica dell'adattamento. La totalità dei test parametrici effettuati accetta l'ipotesi nulla con un grado di significatività pari a 0.05.

6.3.6 Fase di Data Manipulation dei nodi

Si è scelto di analizzare anche l'andamento degli errori sull'intero log considerando la tipologia del nodo che ha generato tale entry

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

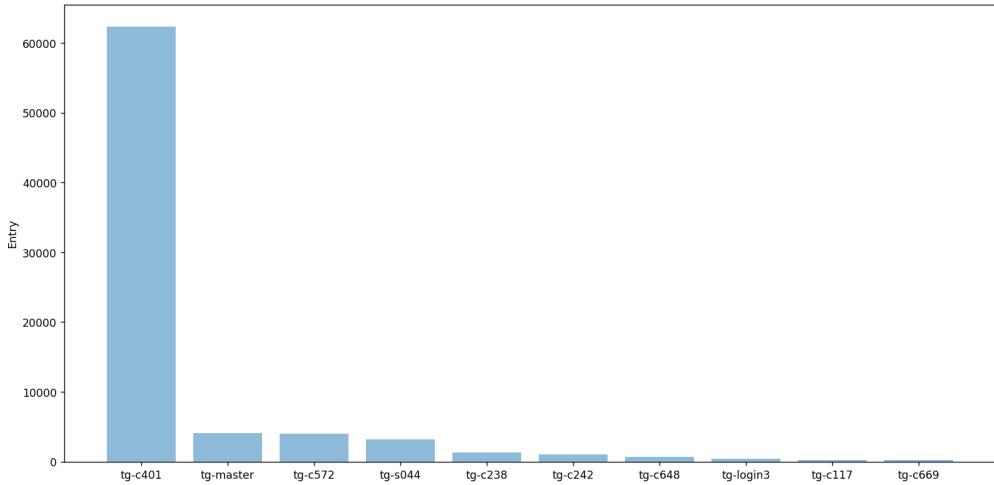


Figura 6.19: Mercury Node Error Count

Per verificare la tipologia di nodo che concorre al maggior numero di entry, si è scelto di **raggruppare i nodi per tipologia**. Come si evince dalla figura, i nodi di tipo **computazionale** sono i nodi che concorrono alla maggior parte degli errori

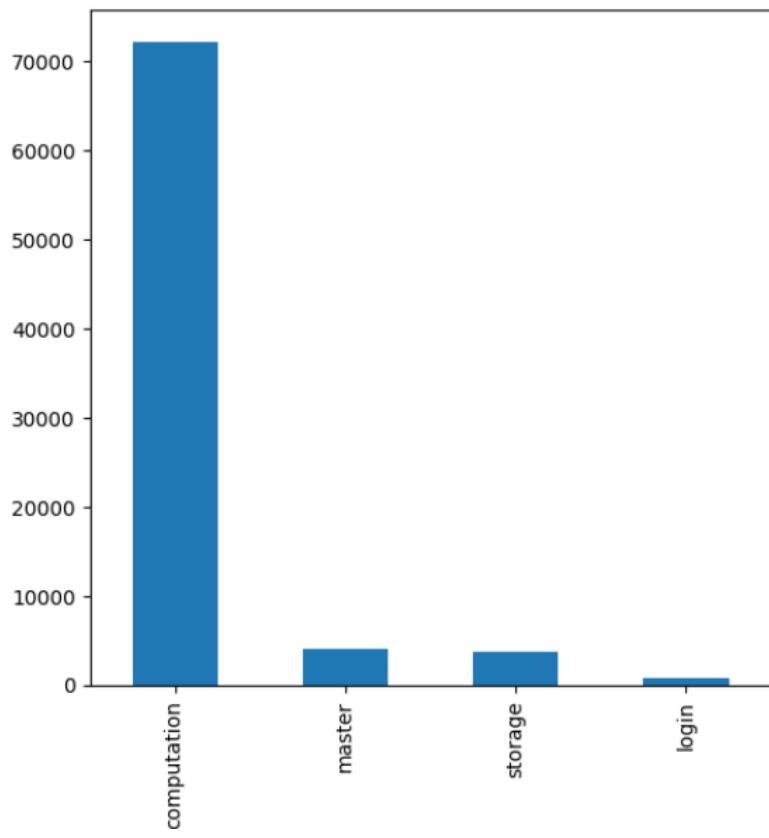


Figura 6.20: Mercury Error per Node Type

Sensitivity Analysis

A valle di tale filtraggio, è stata eseguita la fase di analisi per trovare la migliore finestra di coalescenza per ogni tipologia di nodo.

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

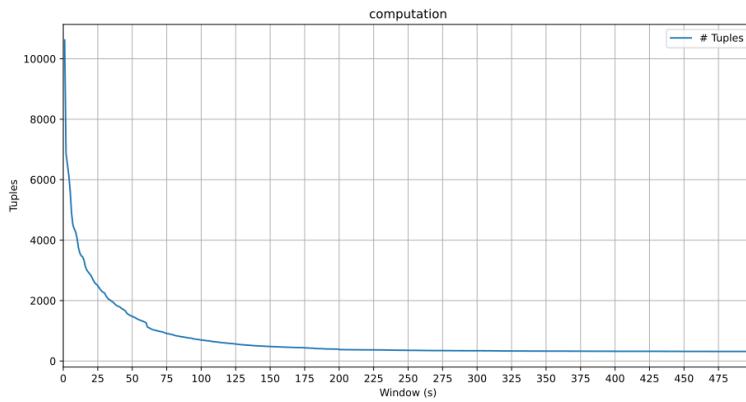


Figura 6.21: Mercury Type Computation Sensitivity

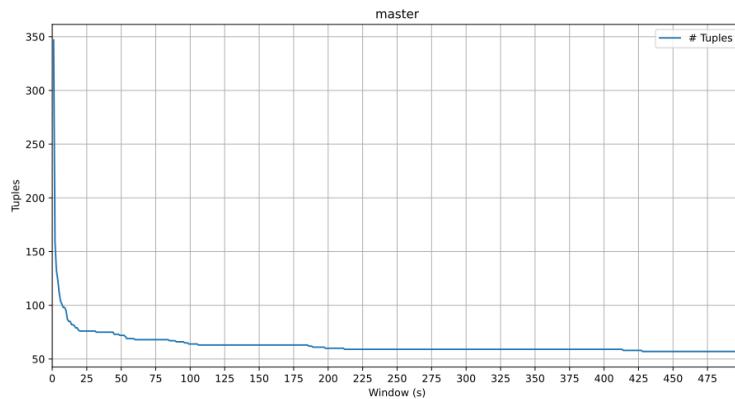


Figura 6.22: Mercury Type Master Sensitivity

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

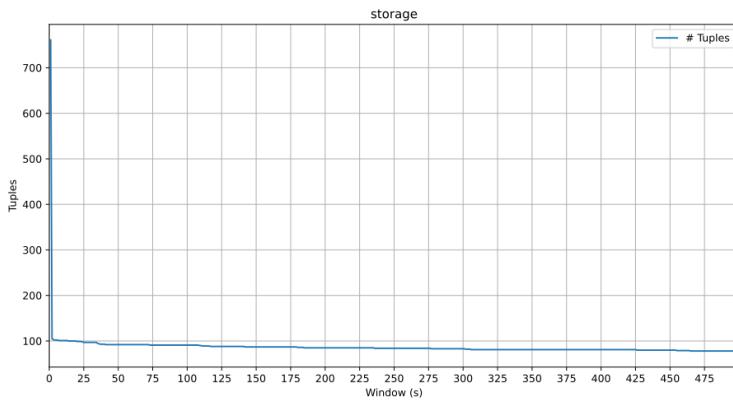


Figura 6.23: Mercury Type Storage Sensitivity

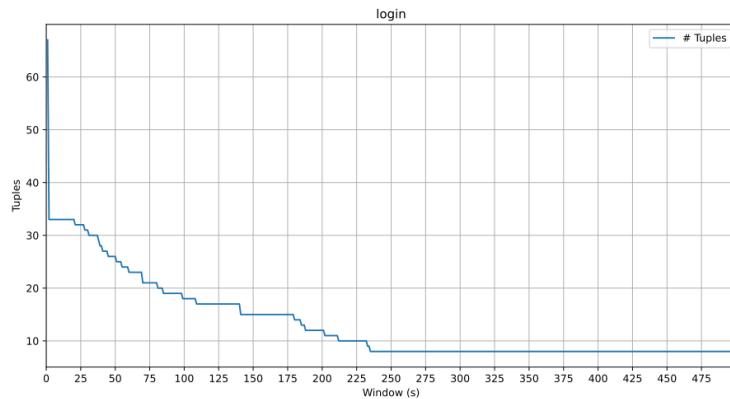


Figura 6.24: Mercury Type Login Sensitivity

Tipologia nodo	Finestra
computation	200
master	150
storage	100
login	300

6.3.7 Fase di Data Analysis dei nodi

Scelte le finestre di coalescenza, si è eseguito l'algoritmo di tupling, e tramite Matlab si è graficato l'andamento delle **curve reliability empiriche ottenute** e dei rispettivi fitting

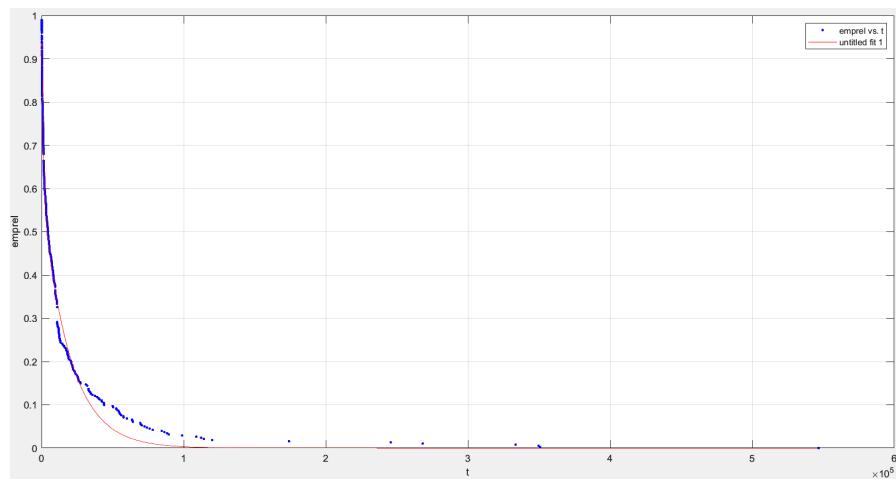


Figura 6.25: Mercury Type Computation Fit

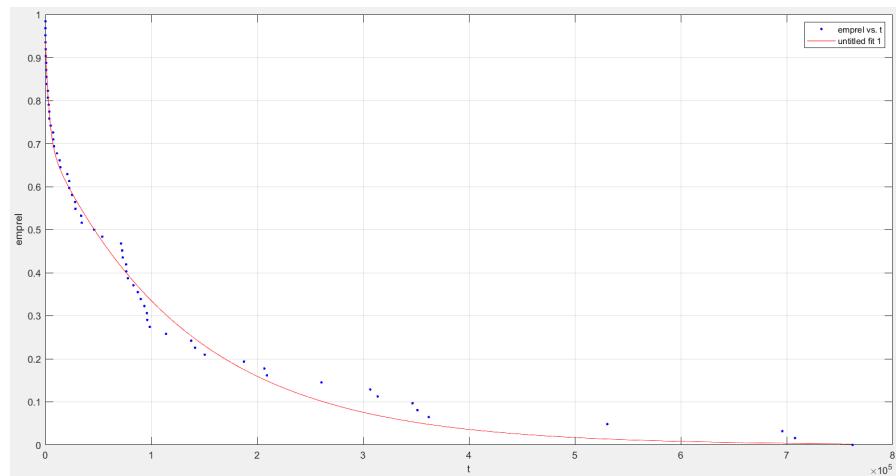


Figura 6.26: Mercury Type Master Fit

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

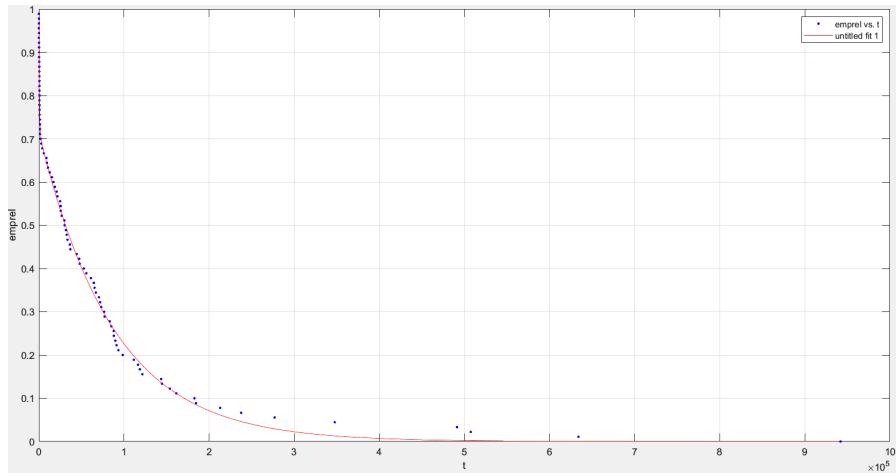


Figura 6.27: Mercury Type Storage Fit

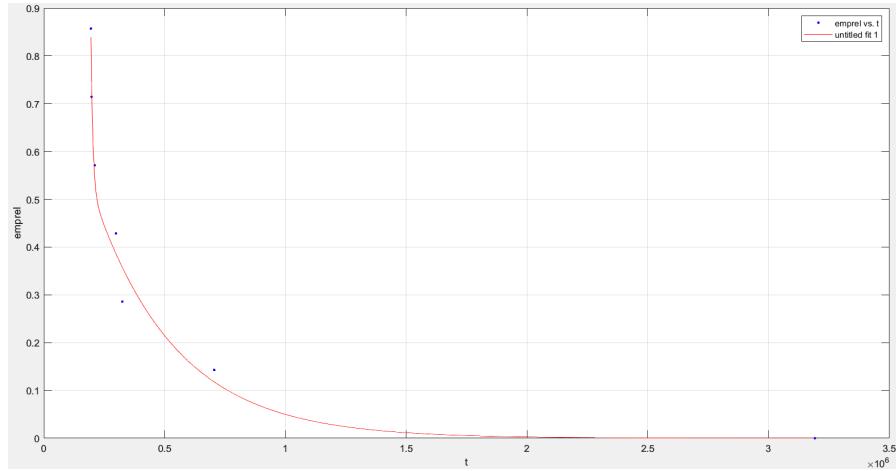


Figura 6.28: Mercury Type Login Fit

	Rel teorica	H-val	p-val	k-val
computation	Exp2	0	0.5381	0.0603
master	Exp2	0	0.9839	0.0806
storage	Exp2	0	1	0.0333
login	Exp2	0	1	0.1429

A valle dell'ottenimento di tali distribuzioni teoriche, si vanno ad effettuare i test per la verifica dell'adattamento. La totalità dei test parametrici effettuati

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

accetta l'ipotesi nulla con un grado di significativit'a pari a 0.05. Confrontando le reliability è possibile affermare che i nodi computazionali sono i nodi che hanno la peggiore reliability all'interno del sistema Mercury, mentre i nodi di login espongono la reliability migliore. In figura gli andamenti a confronto

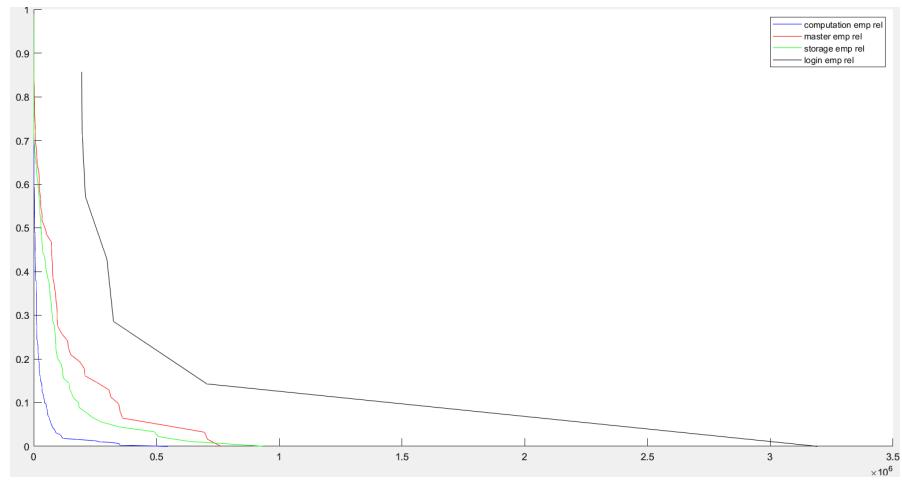


Figura 6.29: Confronto curve reliability empiriche

Inoltre si osserva che la curva empirica di reliability della tipologia computazional è simile alla curva empirica di riability del sistema, allora possiamo confermare l'ipotesi della stretta dipendenza della reliability del sistema da questa tipologia di nodi.

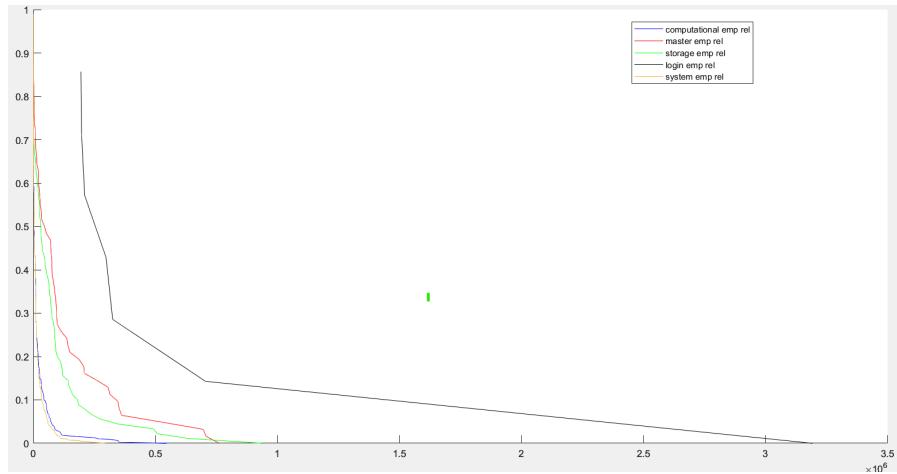


Figura 6.30: Confronto con curva sistema

Approfondimento nodo tg-c401

Per verificare se effettivamente il nodo rappresenta un **bottleneck** nel sistema, si procede **effettuando le operazioni di data manipulation e data analysis dei dati di tale nodo al fine di calcolare la reliability empirica**. La finestra di coalescenza selezionata è pari a 200, si ottengono 59 tuple.

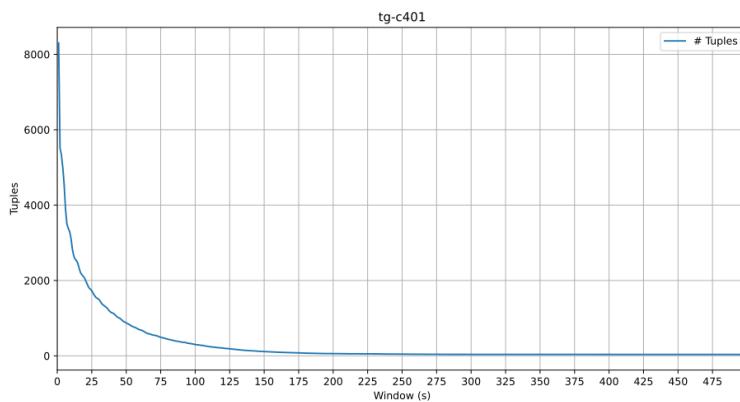


Figura 6.31: Mercury tg-c401 Computation Sensitivity

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

Trovata la finestra di coalescenza ottimale, si passa all'esecuzione dello script di tupling, e alla costruzione dei grafici delle curve empiriche, mostrati in figura.

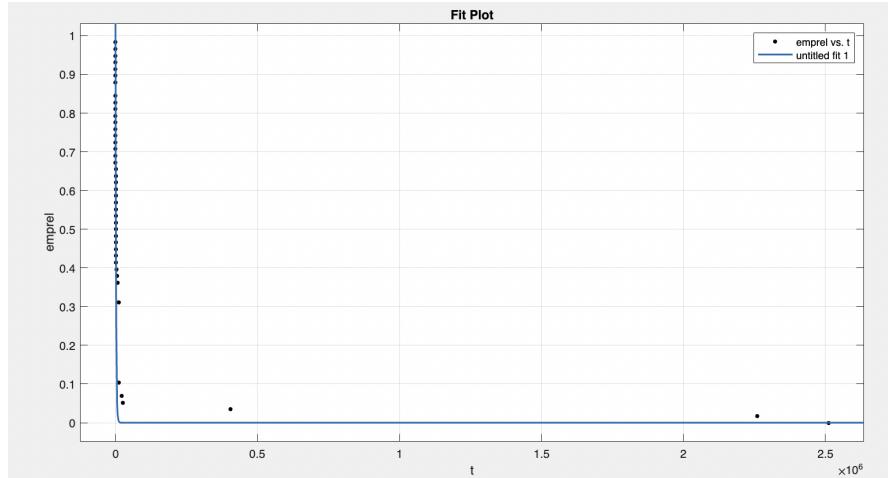


Figura 6.32: Mercury tg-c401 Empirical Reliability

H-value	p-value	k-value
0	0.1687	0.2326

La curva di reliability mostra un brusco calo per valori di tempo bassi, e confrontata con la curva di reliability dell'intero sistema, si nota che per un certo valore di tempo si ha che il nodo ha una reliability peggiore dell'intero sistema. Questo di fatto conferma l'ipotesi di bottleneck del nodo c401.

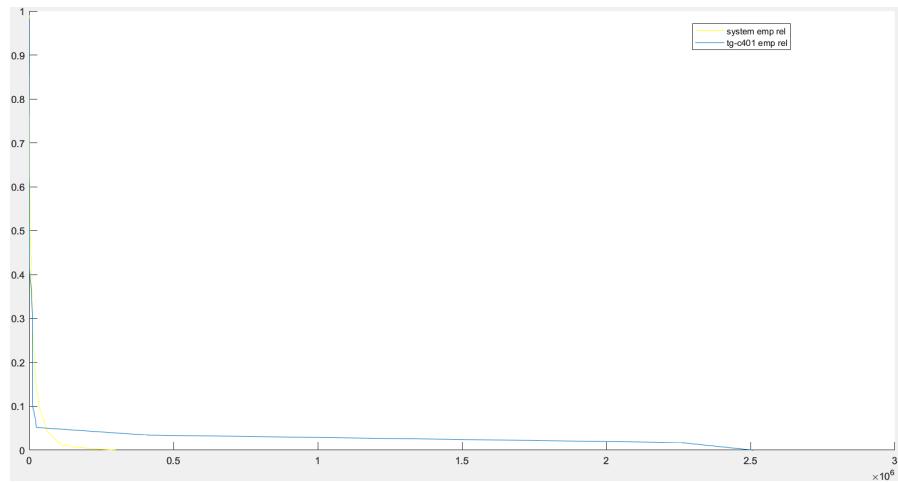


Figura 6.33: Mercury tg-c401 vs system

6.3.8 Domande

Si possono usare le stesse finestre di coalescenza in diversi nodi e in diverse categorie di nodi?

Non si può definire la stessa finestra per i diversi nodi e categorie di nodi, dato che si avrebbero stime troppo pessimistiche in alcuni casi, con alta probabilità di incorrere in troncamenti.

Cosa si può dire della relazione tra l'affidabilità del sistema e l'affidabilità dei nodi?

I nodi da cui dipende strettamente la reliability del sistema sono quelli computazionali, dato che essi hanno una curva di reliability simile a quella del sistema totale. Il sistema presenta una reliability peggiore di ogni tipologia di nodi, quindi non è possibile dire con certezza quale categoria di nodi sia problematica.

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

Esistono colli di bottiglia dell'affidabilità (cioè, i principali contributori al numero totale di fallimenti)?

Il nodo c401 probabilmente rappresenta un collo di bottiglia per l'affidabilità avendo esso una reliability peggiore del sistema stesso per un certo tempo t . Ciò conferma è confermato anche dal numero alto di entry presenti nel log di errore appartenenti al nodo tg-c401.

Nodi funzionali simili (per esempio, due nodi di calcolo) presentano parametri di affidabilità simili?

Per confrontare le reliability di nodi simili si è scelto di considerare tre nodi computazionali, in particolare il nodo c401, c238 e c324, i primi due sono tra i primi 5 nodi per numero di entries nel log di errore, mentre l'ultimo presenta meno entries.

Di seguito si riportano le curve di sensitività e la tabella riepilogativa:

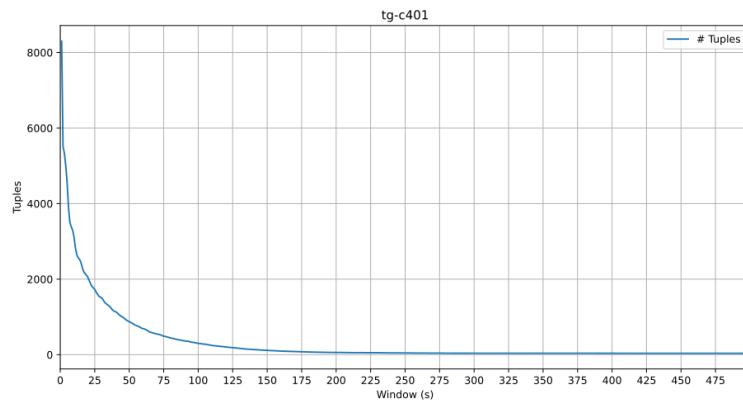


Figura 6.34: Mercury tg-c401 Sensitivity

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

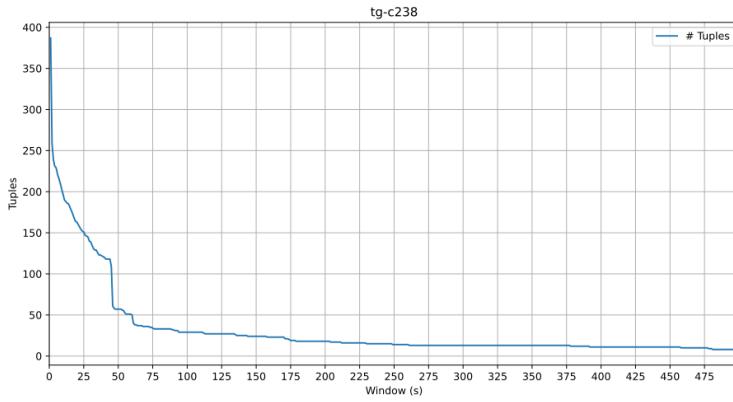


Figura 6.35: Mercury tg-c238 Sensitivity

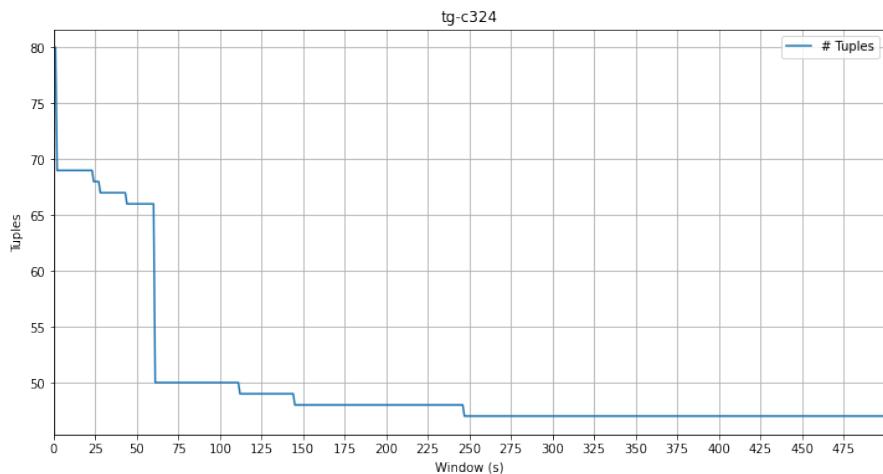


Figura 6.36: Mercury tg-c324 Sensitivity

Nodo	Entries	Finestra	Tuple
tg-c401	62340	200	59
tg-c238	1273	200	18
tg-c324	240	200	48

Al fine di effettuarne il confronto, si riportano in figura 6.37 le rispettive empirical reliability

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

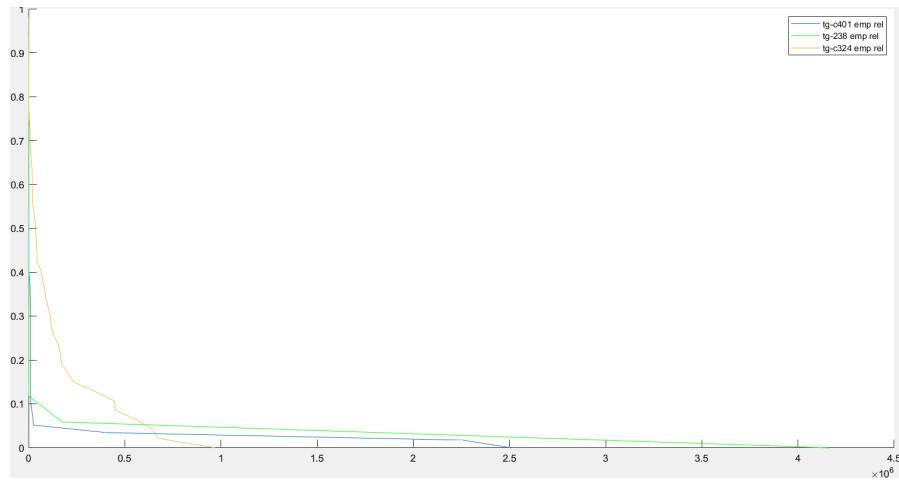


Figura 6.37: Confronto curve di empirical reliability

In definitiva, si evince dal confronto delle curve che i due nodi tg-c401 e tg-c238 presentano una reliability simile, mentre il nodo tg-c324 presenta una reliability nettamente maggiore.

Esiste una relazione tra i tipi di errore e il nodo?

Raggruppando per tipologia di nodo, è possibile vedere che a nodi computazionali corrisponde la maggior parte di errori di tipo DEV e MEM, mentre per altre tipologie di nodi si ha una distribuzione uniforme. Gli errori presenti nei nodi di tipo storage e login riguardano per la maggior parte errori di tipo I-O, invece i nodi di tipo master generano errori di tipo NET.

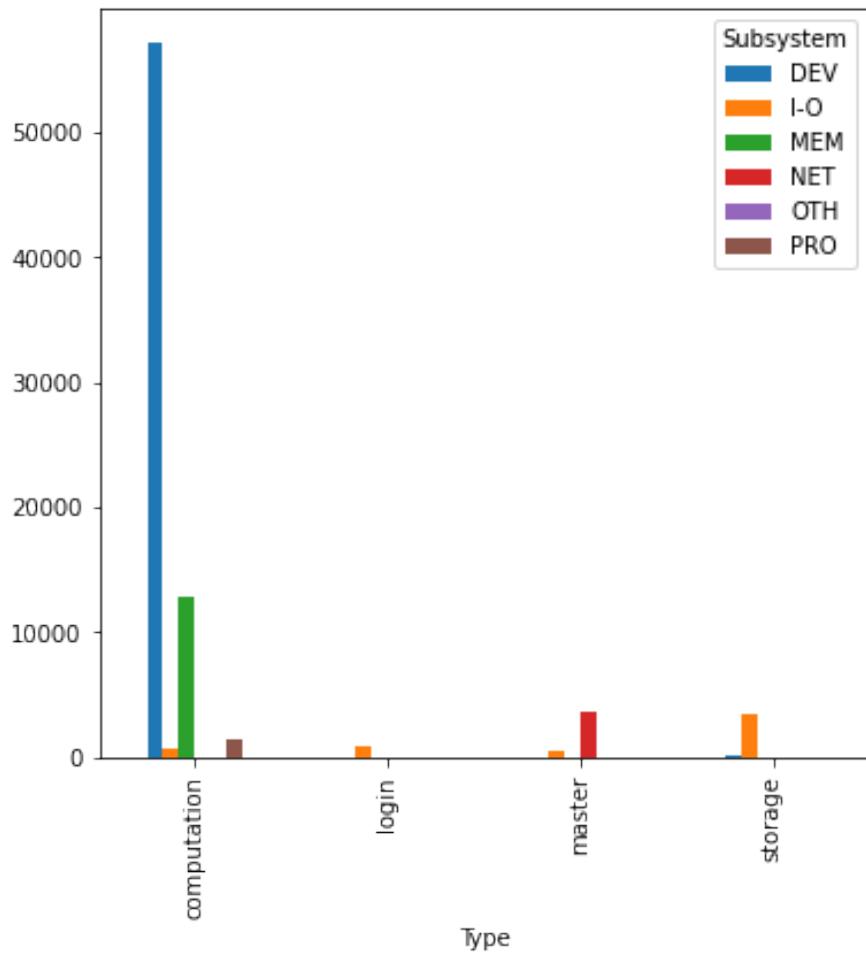


Figura 6.38: Mercury Distribuzione Errori

6.4 Blue Gene/L

6.4.1 Descrizione del sistema

Blue Gene/L (BGL) è un supercomputer sviluppato da IBM. Come riportato in figura, il sistema è costituito da 64 rack (o cabinet), ognuno dei quali è diviso in 2 midplane; ciascun midplane contiene 16 node board (32 per rack), ognuna delle

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

quali contiene a sua volta 16 compute card e 2 I/O card opzionali. I nodi dotati di una card di I/O sono N0, N4, N8 ed NC.

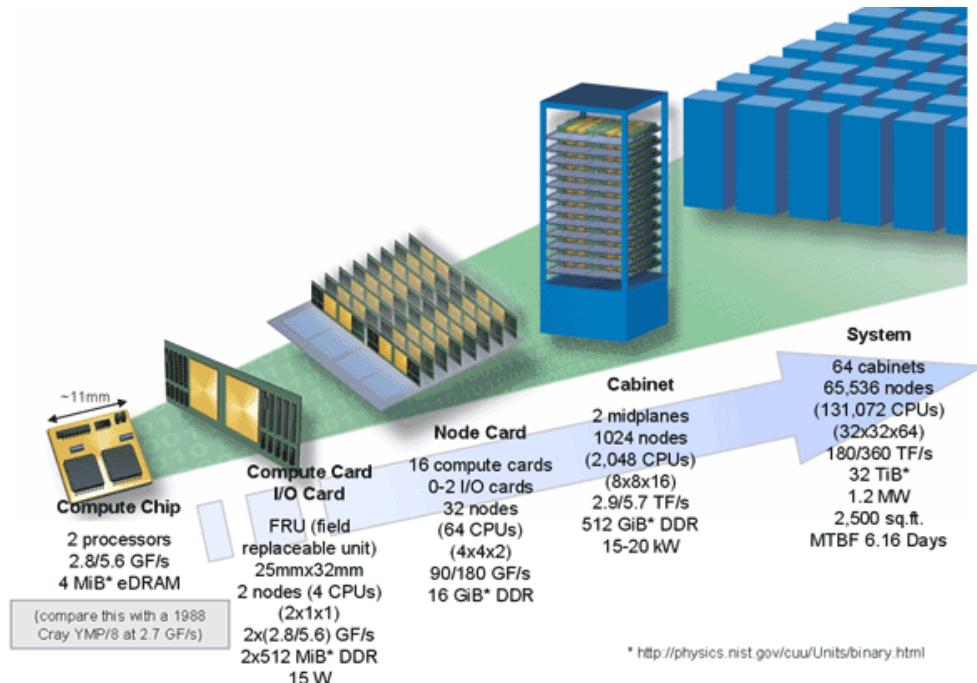


Figura 6.39: Architettura Blue Gene/L

I dati da analizzare relativi a BGL sono contenuti in un file di log chiamato BGLErrorLog.txt. Il file contiene 125624 entry, ognuna delle quali prevede i seguenti campi:

- **timestamp:** intero in formato UNIX
- **nodo di origine:** stringa del tipo Rxx-Mx-Nx indicante il **nodo** in cui si è verificato l'errore. In particolare, sono indicati **rack(R)**, **midplane(M)** e **nodo(N)** del sistema che ha generato l'evento

- **card di origine**: stringa del tipo Jxx-Uxx indicante la card del nodo da cui ha origine l'errore. È possibile dividere le card in due grandi categorie, **card di I/O** (J18-Uxx) e **card di computation**
- **messaggio testuale**: stringa indicante il messaggio di errore

6.4.2 Fase di Data Manipulation del sistema

Come fatto per il sistema Mercury, si vanno a definire le tuple al variare della finestra di coalescenza, per poi decidere la finestra di coalescenza ottimale ed effettuare il tupling.

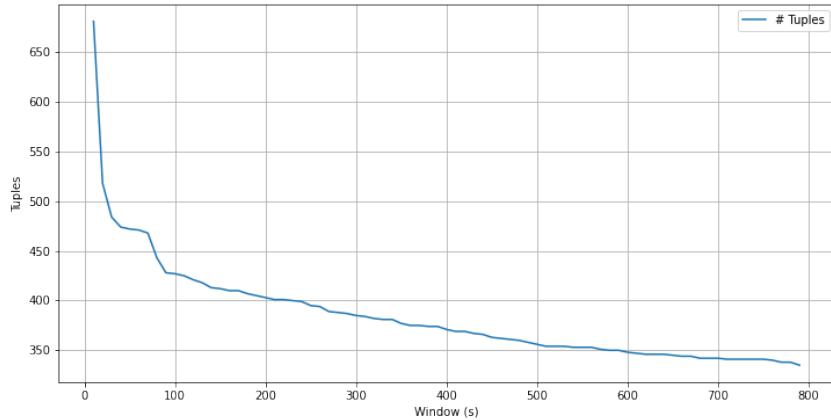


Figura 6.40: Blue Gene System Sensitivity

Il punto di knee della curva è intorno al valore di finestra pari a 150, quindi si decide di utilizzare 200 come finestra di coalescenza, come accade per Mercury.

Bottleneck

Si analizza la distribuzione di errori per ogni tupla, in modo da individuare intervalli temporali dove sono presenti la maggior parte degli errori.

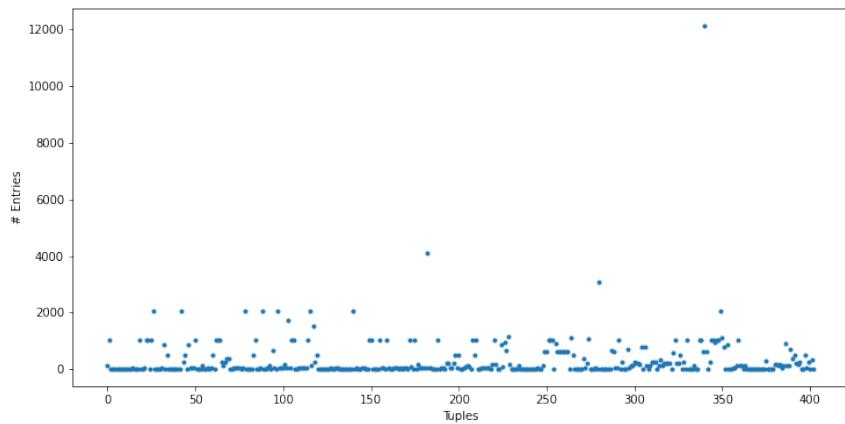


Figura 6.41: Blue Gene System Scatter Tuple

La distribuzione risulta più omogenea, non è possibile quindi individuare un particolare intervallo temporale di interesse. Questo suggerisce che gli errori registrati nel file di log sono dispersi più uniformemente nel tempo, rispetto al sistema precedente. E' possibile effettuare un'analisi più approfondita per individuare eventuali bottleneck, infatti è possibile separare le due tipologie di errori rispettivamente dovuti alle card di I-O oppure relativi alle card di computazione.

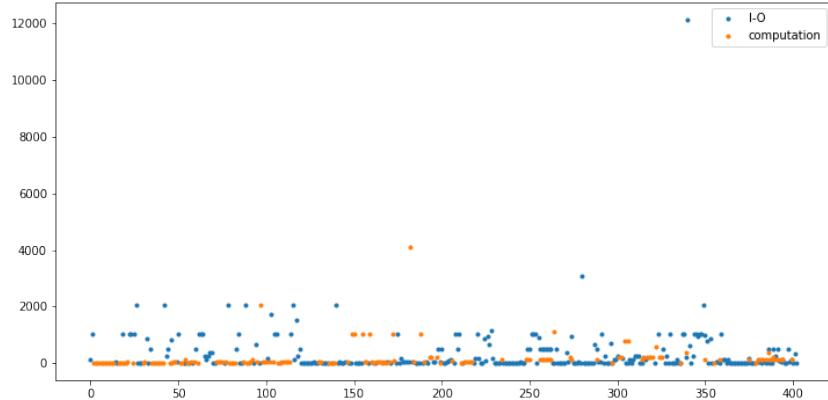


Figura 6.42: Blue Gene System Scatter Tuple CardType

Una conclusione che è possibile trarre da tale analisi è che la maggioranza delle entry del log appartengono ad errori classificati come di tipo I-O. Siccome l’omogeneità dei due tipi di entry nelle varie tuple non permette di evidenziare alcun colpevole principale, è ragionevole pensare che un ipotetico bottleneck sia l’intero sistema di I-O di BGL. Infatti, le entry relative a operazioni I-O sono 99987 circa il 79.59% del file di log, mentre quelle computazionali sono 25637 circa il 20.41%.

Analisi Troncamenti

Come per Mercury si calcola il valore q corrispondente al 10° percentile(0.1 quantile) dei tempi di interarrivo delle tuple. Tale valore è circa 451 secondi(circa 8 minuti), si individuano poi tutte le coppie di tuple adiacenti che distano meno di questo valore. Per ognuna di queste coppie è possibile stimare un troncamento, da questa prima stima si ottengono 41 troncamenti, il che significa che per circa il 10% delle tuple totali potrebbe essersi verificato un troncamento. Il numero di

troncamenti risulta estremamente basso, sintomo di una buona scelta della finestra di coalescenza.

Analisi Collisioni

Analogamente si vuole effettuare una stima delle collisioni commesse a causa del tupling. Il numero di potenziali collisioni è ottenuto individuando il numero di tuple per le quali si hanno piu' nodi di origine. Sono quindi stimate 185 collisioni, il che significa che per circa l'46% delle tuple totali potrebbero essersi verificate collisioni. La percentuale di collisioni è elevato, si ricalcolano le collisioni contando il numero di tuple per le quali si hanno più tipologie di card differenti(I-O o computational). Seguendo tale approccio si sono stimate 39 collisioni, il che significa che per circa il 10% delle tuple totali potrebbero essersi verificate collisioni. Il numero di collisioni risulta quindi accettabile, ciò conferma la scelta della finestra di coalescenza per Blue Gene.

6.4.3 Fase di Data Analysis del sistema

Si passa allo studio della reliability empirica, tramite lo stesso script usato per Mercury. Una volta calcolata la curva di reliability empirica è possibile testare la bontà di adattamento alla curva a varie distribuzioni, tramite lo strumento di curve fitting[matlab]. La curva di fitting migliore è l'esponenziale di secondo grado, con i seguenti parametri:

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

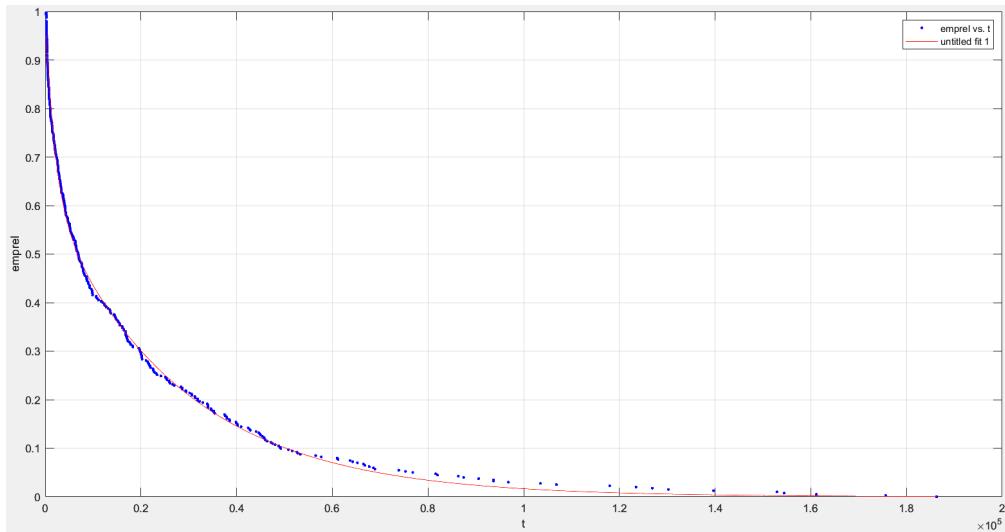


Figura 6.43: Blue Gene System Empirical Reliability Fitting

$$\text{fit-BGL}(x) = ae^{bx} + ce^{dx}$$

a	b	c	d
0.3675	-0.0004837	0.6222	-3.708e-05

Per confermare la bontà del fitting possiamo osservare i parametri di GOF, infatti si ottengono un SSE prossimo allo zero ed un R-square vicino ad 1:

- SSE: 0.0693
- R-square: 0.9978

Per avere una conferma definitiva si esegue il test non parametrico di Kolmogorov-Smirnov; i risultati sono riportati in tabella.

H-value	p-value	k-value
0	0.8954	0.0406

Dai risultati del test in tabella si nota come è possibile accettare l'ipotesi nulla e confermare la bontà dell'adattamento della reliability empirica ad una reliability

esponenziale. Come evidenziato per il sistema Mercury, anche il supercomputer BGL presenta una reliability empirica di tipo esponenziale, e quindi i suoi guasti possono essere ricondotti a tipi di fallimenti hardware, esattamente come avveniva per Mercury.

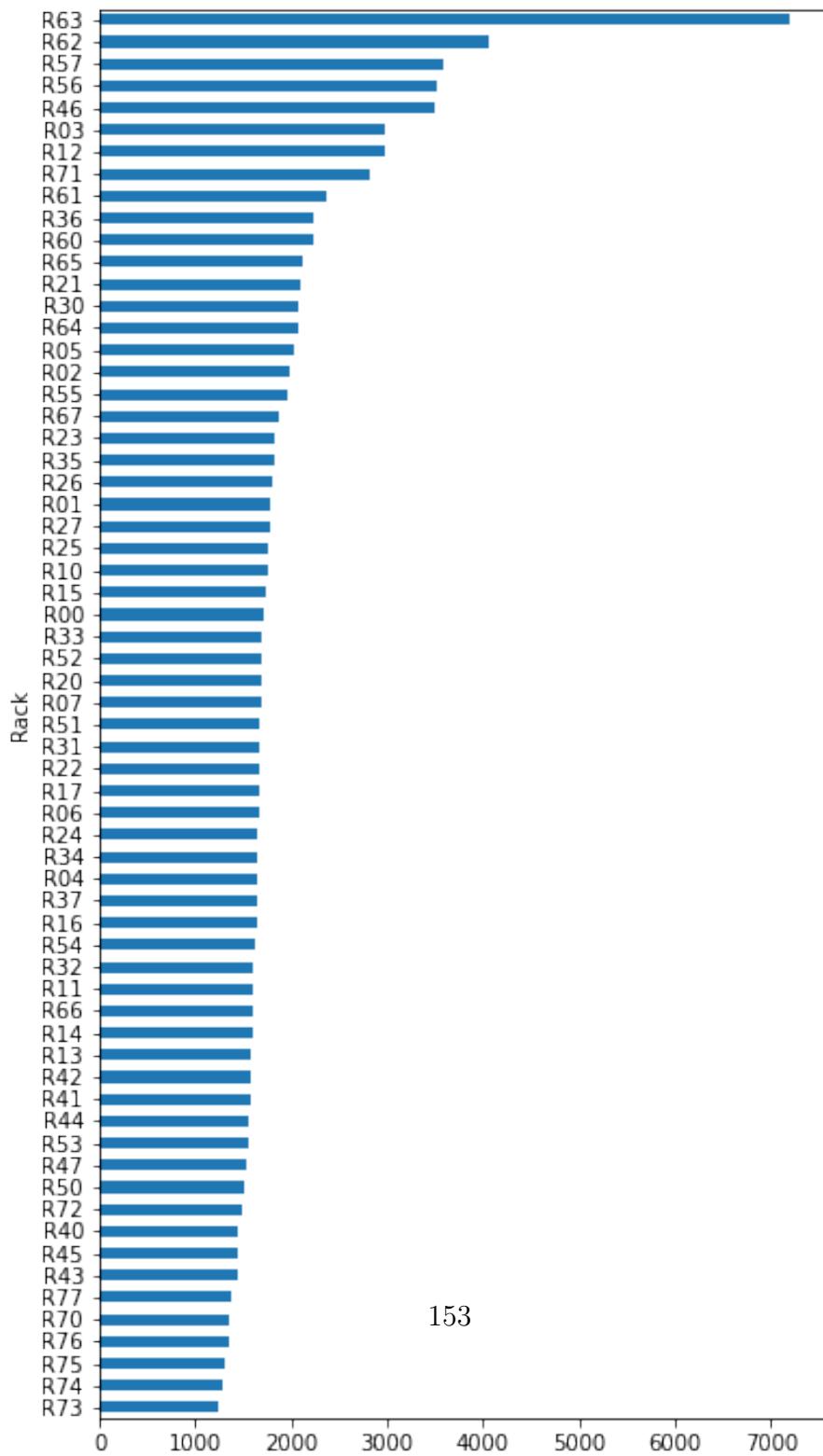
6.4.4 Fase di Data Manipulation dei nodi

Così come fatto per il sistema Mercury, si è deciso di effettuare un analisi dei nodi anche del sistema BGL. Si andranno a studiare i rack più critici del sistema, poi si andrà ad approfondire l'analisi evidenziando errori relativi ai midplane dei rack e ai nodi appartenenti ai vari midplane.

Analisi dei Rack

Si raggruppano le istanze di log per rack, così da evidenziare la distribuzione degli errori rispetto ai rack. Da tale grafico si evidenzia che alcuni rack sono responsabili della maggior parte delle entry del file di log, e tali rack sono R63, R62, R57, R56, R46, R03, R12, R71.

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS



Analisi dei Midplane

Ogni rack è composto da due midplane, indicati nel log come M0 ed M1. Si analizzano i rack più critici individuati in precedenza, separando gli errori anche per midplane.

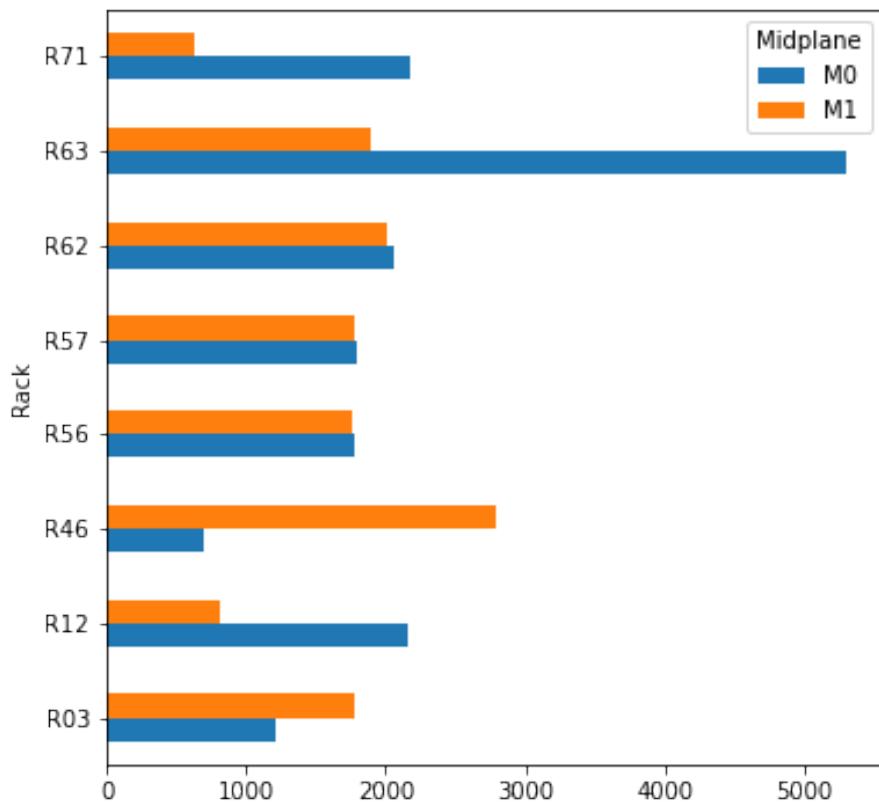


Figura 6.45: Blue Gene Entry per Rack-Midplane

Il rack R-63 presenta più errori sul midplane M0 rispetto ad M1, per gli altri rack la distribuzione è abbastanza bilanciata.

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

Analisi dei Nodi

Si prosegue analizzando la distribuzione delle entry per nodo appartenente ai due midplane dei rack più critici.

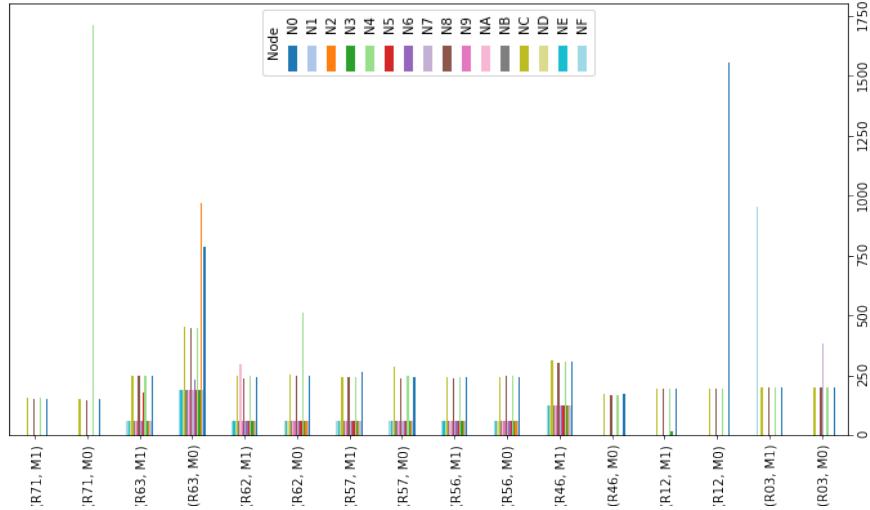


Figura 6.46: Blue Gene Entry per Rack-Midplane-Node

Dalla tabella è possibile trarre alcune conclusioni che riguardano le entry appartenenti ai rack peggiori:

- **Analisi sui rack:** Alcuni rack presentano dei faillimenti bilanciati tra i nodi, in particolare R56,R57,R62 ed R63 rientrano in questi rack. Mentre non è possibile dire lo stesso per gli altri rack. Avendo un bilanciamento di errori tra i nodi, si può dire che tali rack non sono la causa di soli errori di I-O, ma

anche di errori di tipo computazionale. Infine il rack R63 oltre ad essere il rack che presenta più entry nel log presenta un numero alto di errori relativi a tutti i nodi, con un alta percentuale sul midplane M0.

- **Analisi sui nodi:** Tutti gli 8 rack più critici presentano numerosi errori sui 4 nodi dotati di card-IO N0,N4,N8,NC. Per il rack R71 il nodo N4 ha un picco di errori, come per il rack R12 il nodo N0.

Sensitivity Analysis per Rack

Si vuole confrontare la reliability dei rack analizzati, quindi come primo passo si va ad effettuare una analisi di sensitività per decidere la finestra di coalescenza ottimale da utilizzare per il tupling, separando il log di errori per appartenenza rack.

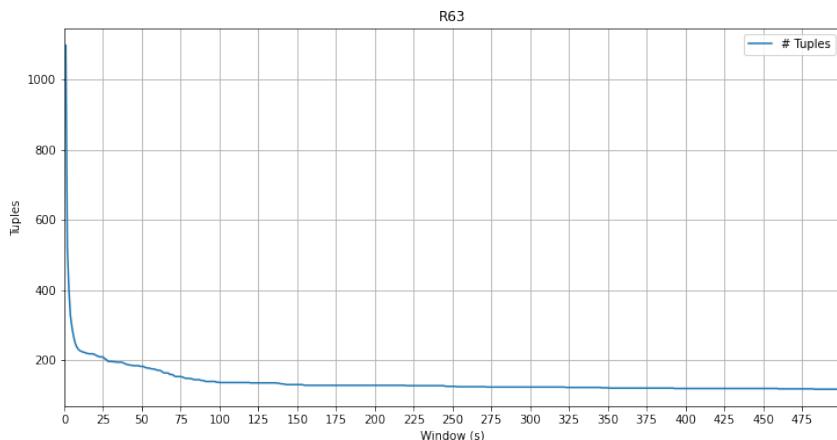


Figura 6.47: Blue Gene R63 Sensitivity

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

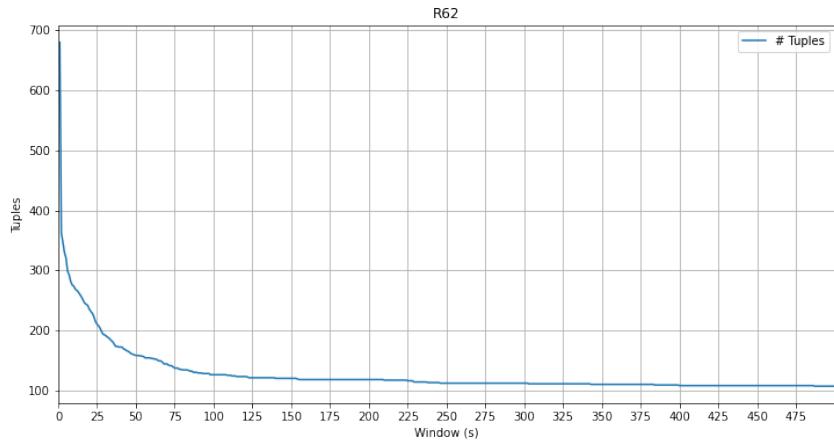


Figura 6.48: Blue Gene R62 Sensitivity

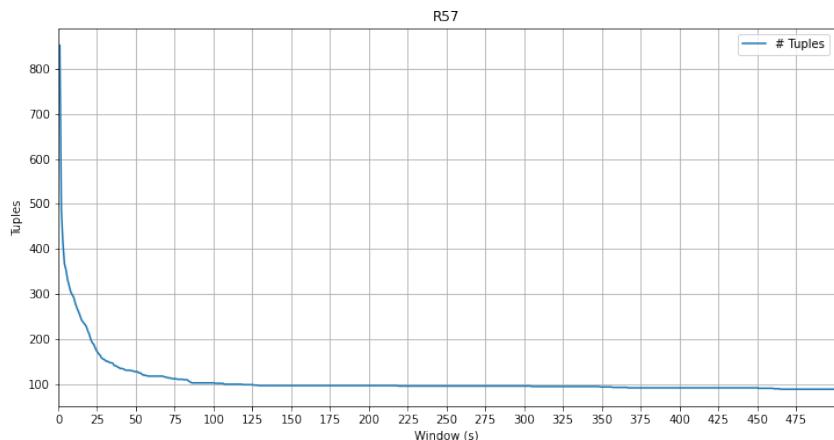


Figura 6.49: Blue Gene R57 Sensitivity

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

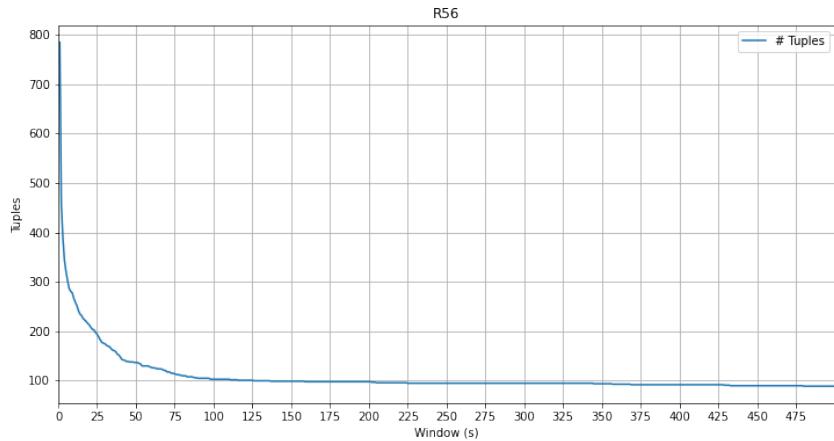


Figura 6.50: Blue Gene R56 Sensitivity

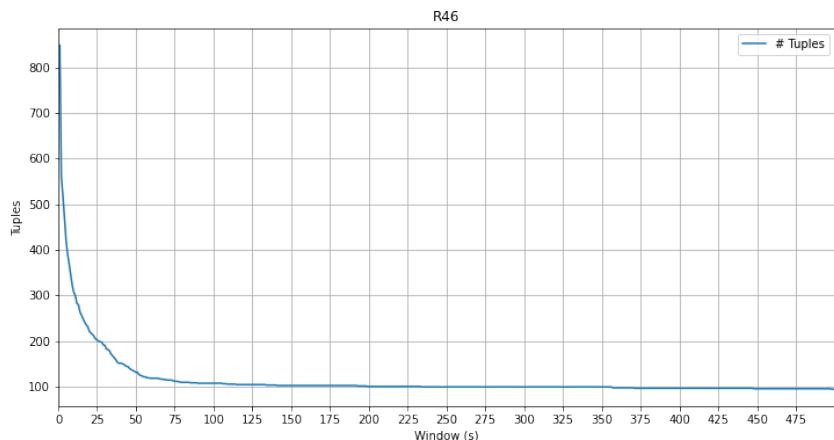


Figura 6.51: Blue Gene R46 Sensitivity

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

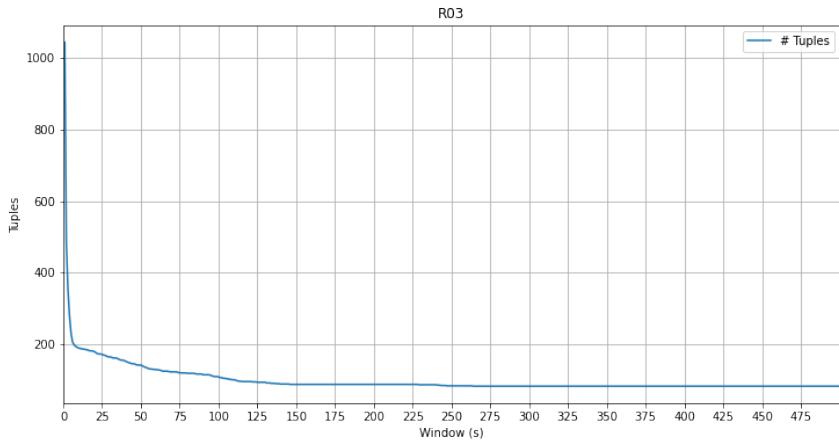


Figura 6.52: Blue Gene R03 Sensitivity

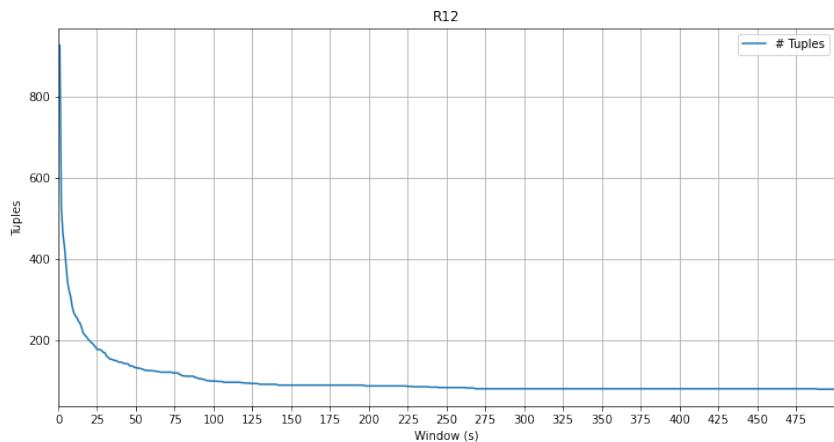


Figura 6.53: Blue Gene R12 Sensitivity

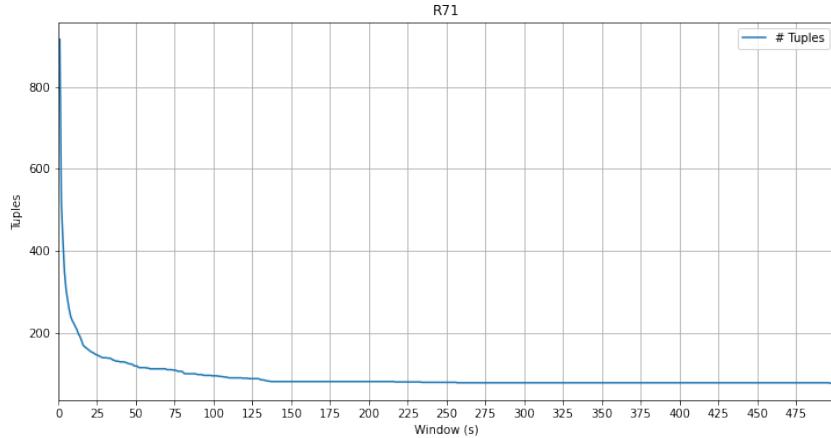


Figura 6.54: Blue Gene R71 Sensitivity

Come si può osservare dai grafici è possibile definire una finestra di coalescenza ideale univoca da utilizzare per tutti i rack, si fissa a 75 secondi, si riporta il numero di tuple ottenute.

Rack	Finestra	Tuple
R63	75	153
R62	75	138
R57	75	111
R56	75	114
R46	75	113
R03	75	120
R12	75	119
R71	75	109

6.4.5 Fase di Data Analysis dei rack

Si passa allo studio della reliability empirica dei vari rack. Una volta calcolate le curva di reliability empirica è possibile testare la bontà di adattamento alla curva a varie distribuzioni, tramite lo strumento di curve fitting[matlab].

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

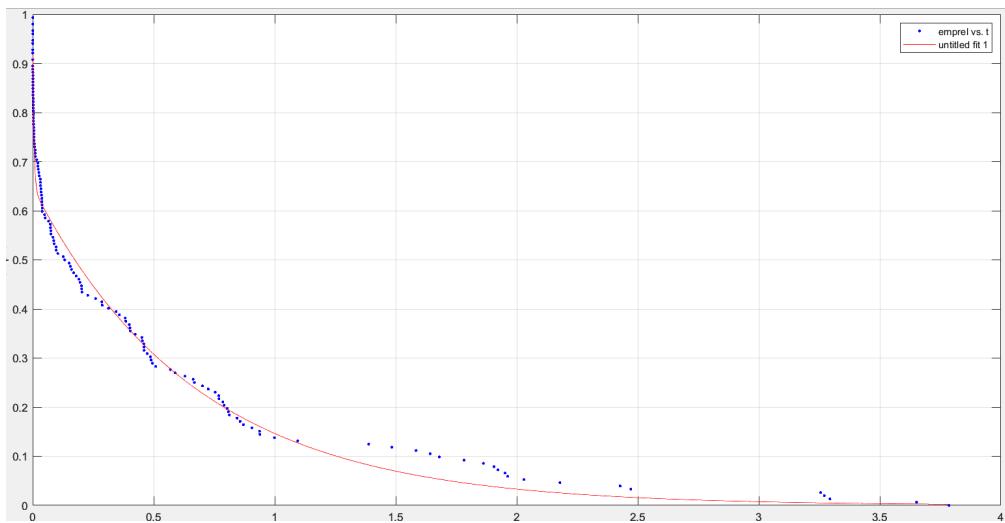


Figura 6.55: Blue Gene R63 Empirical Reliability Fitting

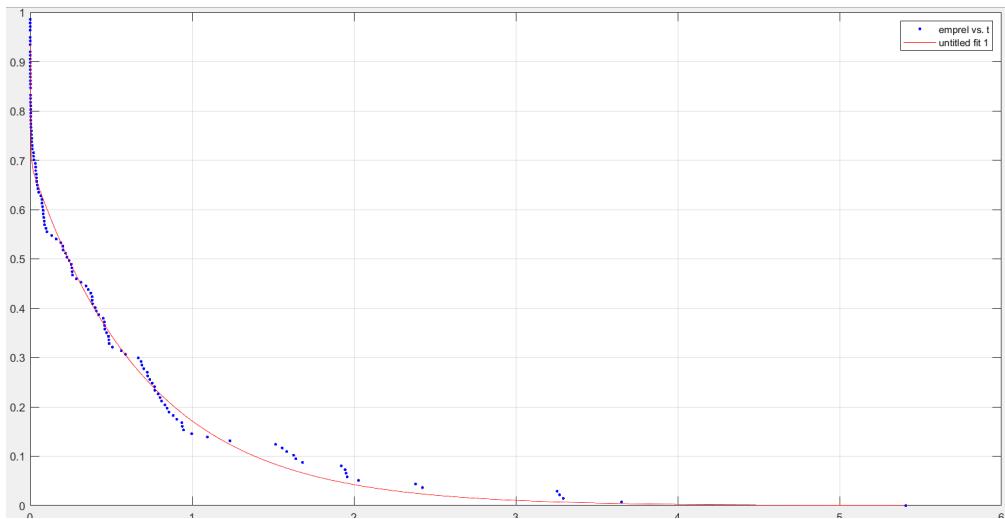


Figura 6.56: Blue Gene R62 Empirical Reliability Fitting

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

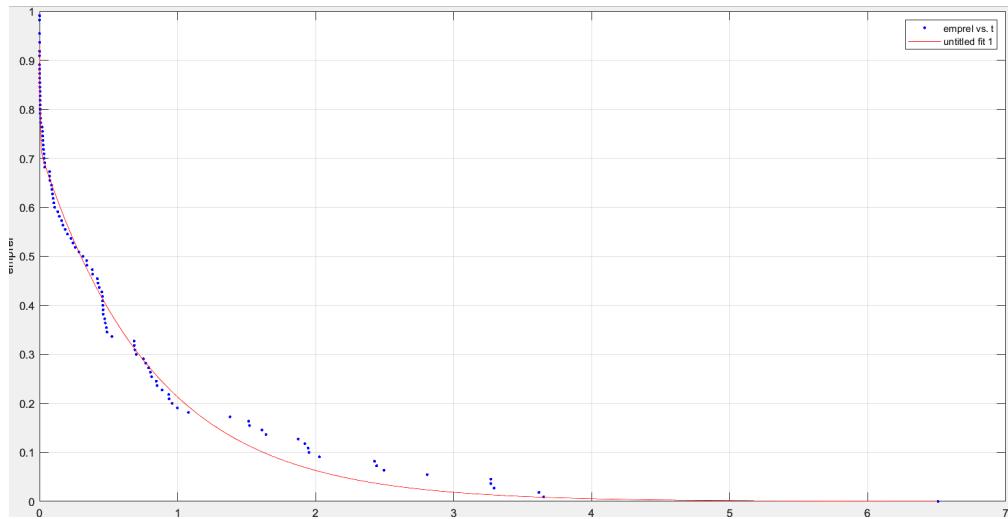


Figura 6.57: Blue Gene R57 Empirical Reliability Fitting

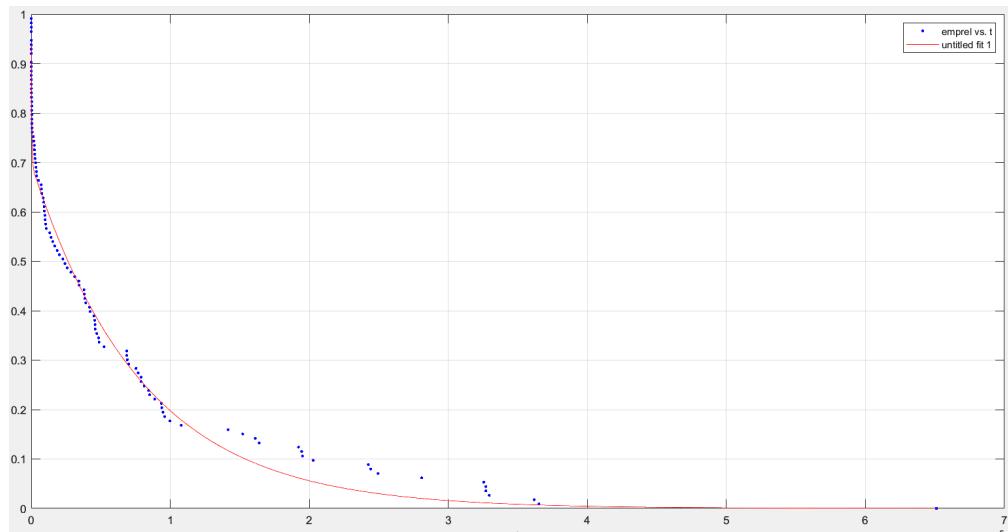


Figura 6.58: Blue Gene R56 Empirical Reliability Fitting

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

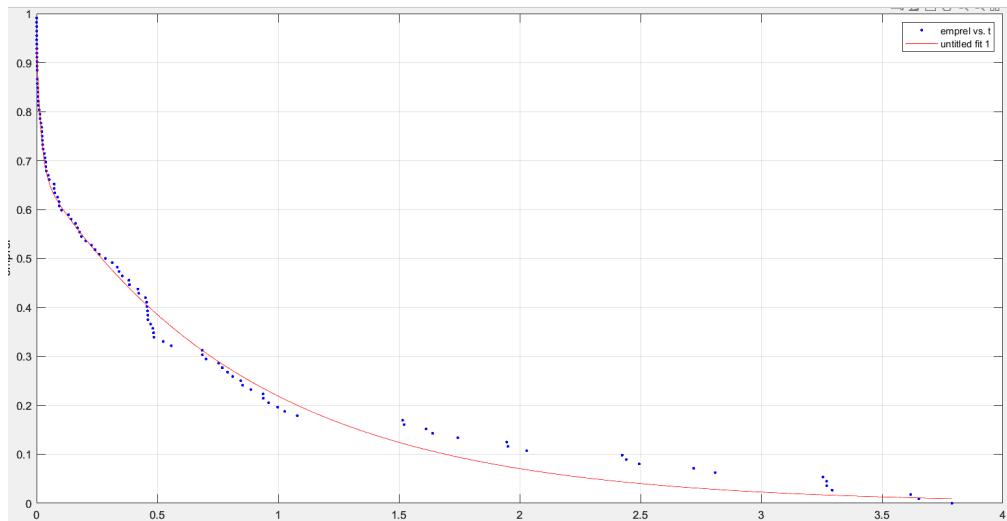


Figura 6.59: Blue Gene R46 Empirical Reliability Fitting

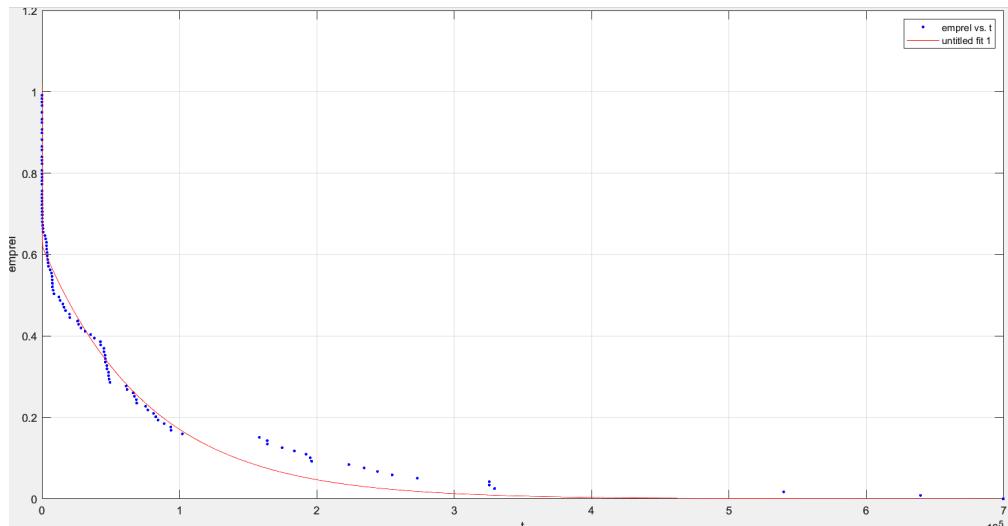


Figura 6.60: Blue Gene R03 Empirical Reliability Fitting

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

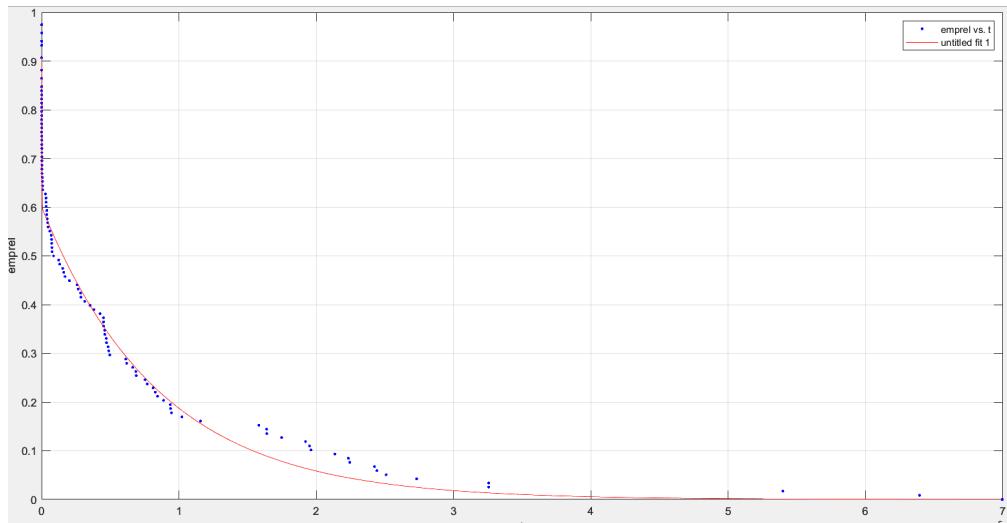


Figura 6.61: Blue Gene R12 Empirical Reliability Fitting

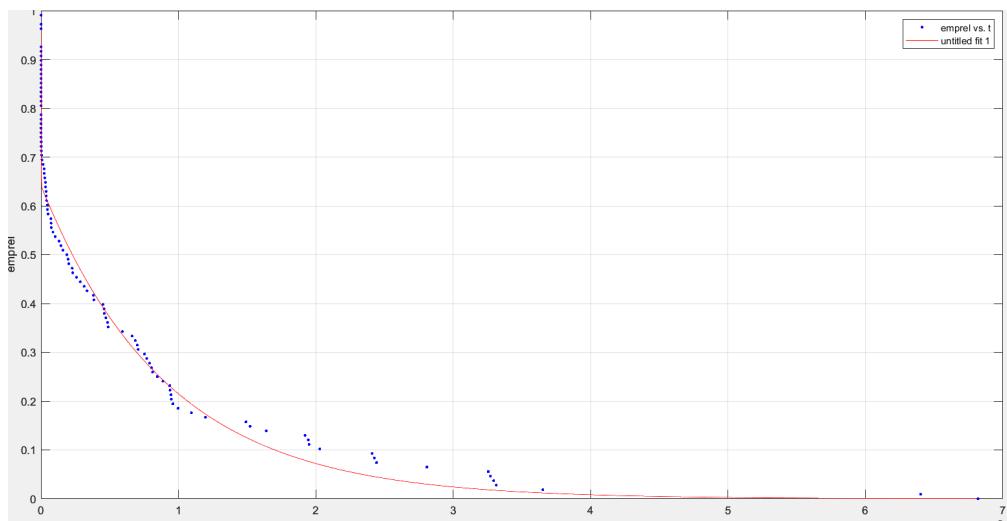


Figura 6.62: Blue Gene R71 Empirical Reliability Fitting

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

Rack	Rel teorica	H-val	p-val	h-val
R63	Exp2	0	0.8717	0.0684
R62	Exp2	0	0.9912	0.0526
R57	Exp2	0	0.9942	0.0571
R56	Exp2	0	0.9261	0.0720
R46	Exp2	0	0.9761	0.0630
R03	Exp2	0	0.8438	0.0810
R12	Exp2	0	0.9727	0.0648
R71	Exp2	0	0.9658	0.0679

A valle dell'ottenimento di tali distribuzioni teoriche, si vanno ad effettuare i test per la verifica dell'adattamento. La totalità dei test parametrici effettuati accetta l'ipotesi nulla con un grado di significatività pari a 0.05.

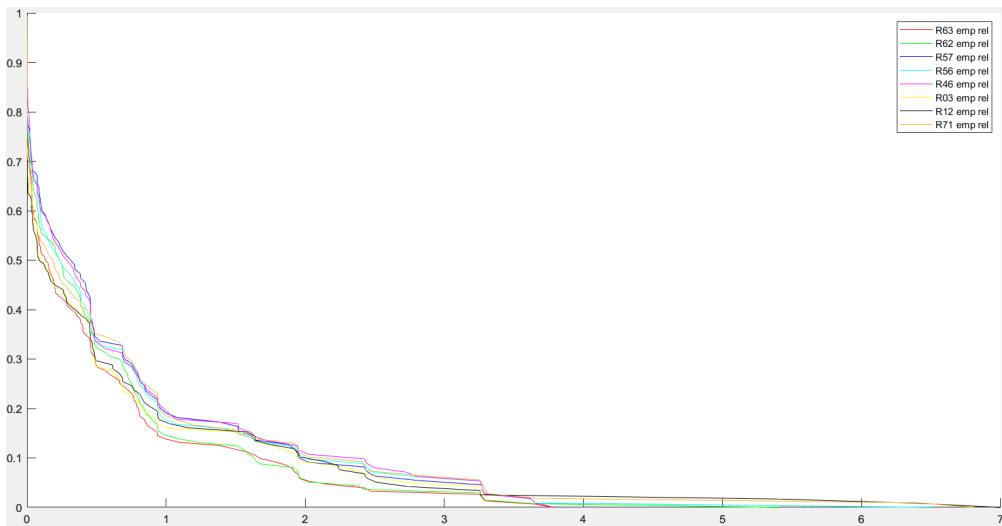


Figura 6.63: Blue Gene Confronto Reliability

Per confrontare tali reliability è utile riportare gli andamenti all'interno di un unico grafico; da tale grafico è possibile verificare come il rack R63 risulti quello a reliability mediamente peggiore, anche se per valori di tempo alti sembrerebbe essere migliore degli altri rack considerati. Inoltre, si osserva che le reliability dei rack non sono molto differenti.

6.4.6 Fase di Data Manipulation delle card

Data l'ipotesi di un ipotetico bottleneck nel sistema di IO e dal confronto dei rack, si è deciso di analizzare la realibility andando a separare i log sul tipo di card computazionale oppure IO. Graficando la distribuzione delle entry così categorizzate si ottiene il seguente istogramma.

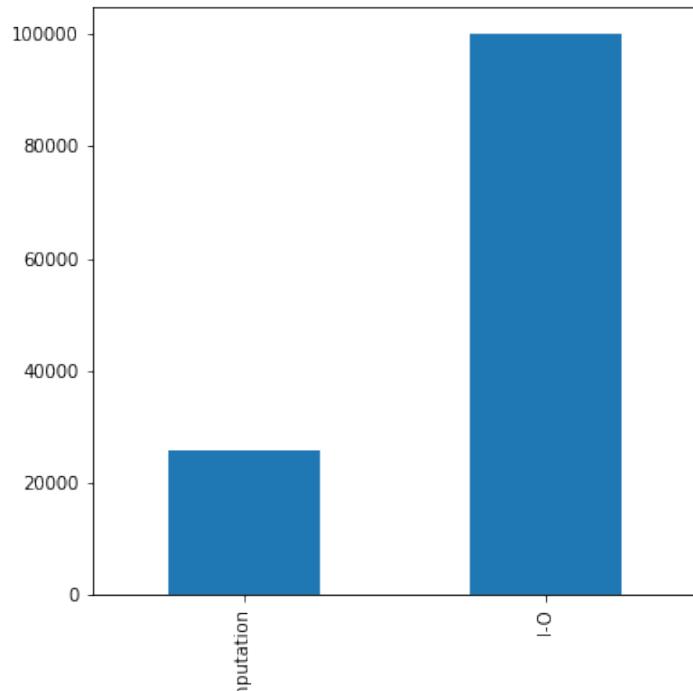


Figura 6.64: Distribuzione Errori per tipo Card

Le card di tipo IO presentano il maggior numero di errori. Raggruppando le entry per Card, si ottiene che le due card di IO(J18-U11, J18-U01) sono responsabili della quasi totalità degli errori nel log. A questo punto è chiaro che il bottleneck del sistema è rappresentato dall'intero sottosistema di I/O.



Figura 6.65: Distribuzione Errori Card
167

Sensitivity Analysis per Card

Si analizzano le curve di andamento delle tuple in funzione della finestra di coalescenza per decidere quali finestre di coalescenza usare.

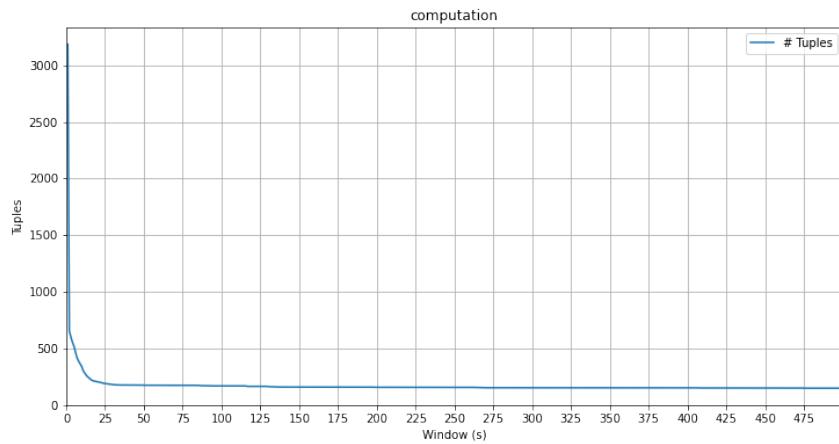


Figura 6.66: Blue Gene Computational Card Sensitivity

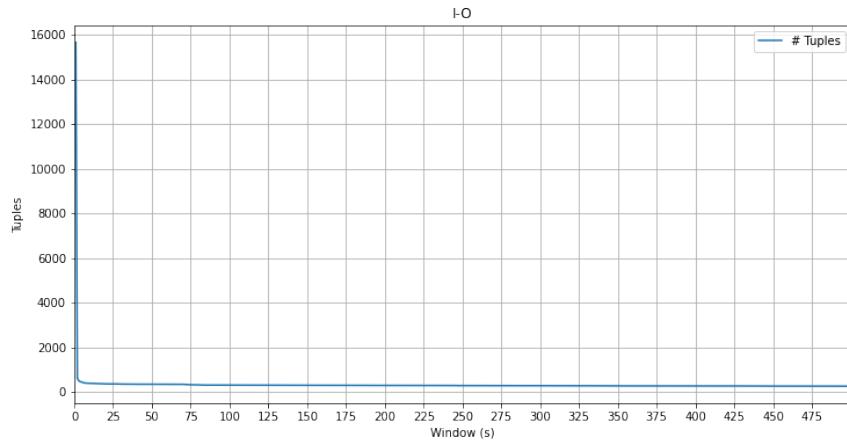


Figura 6.67: Blue Gene IO Card Sensitivity

Per ogni tipologia si scelgono le finestre di coalescenza riportate in tabella assieme al numero di tuple ottenute:

Tipologia	Finestra	Tuple
computational	20	204
IO	10	380

6.4.7 Fase di Data Analysis delle card

Si riportano le due curve di reliability empiriche per le card di tipo computational ed I-O.

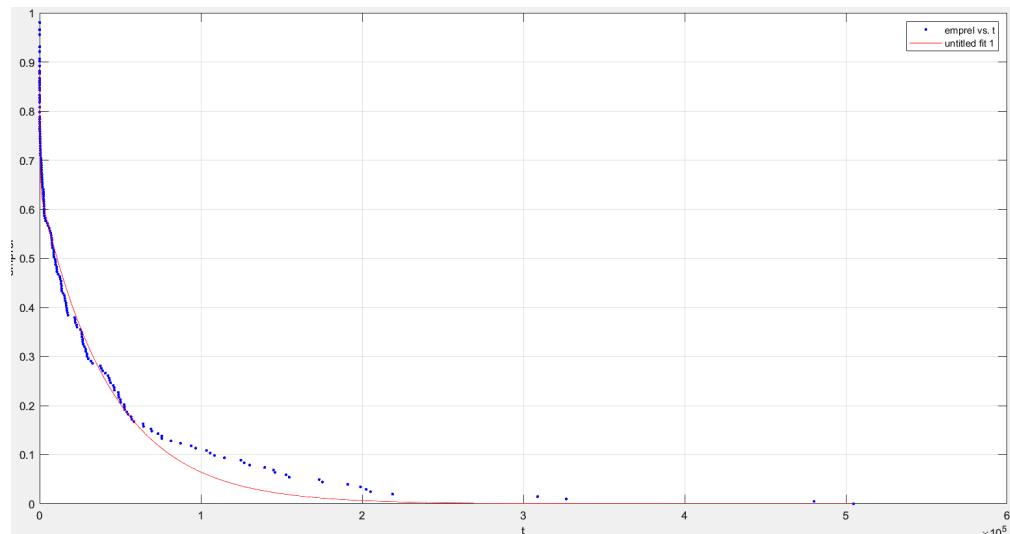


Figura 6.68: Blue Gene Computational Card Empirical Reliability Fitting

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

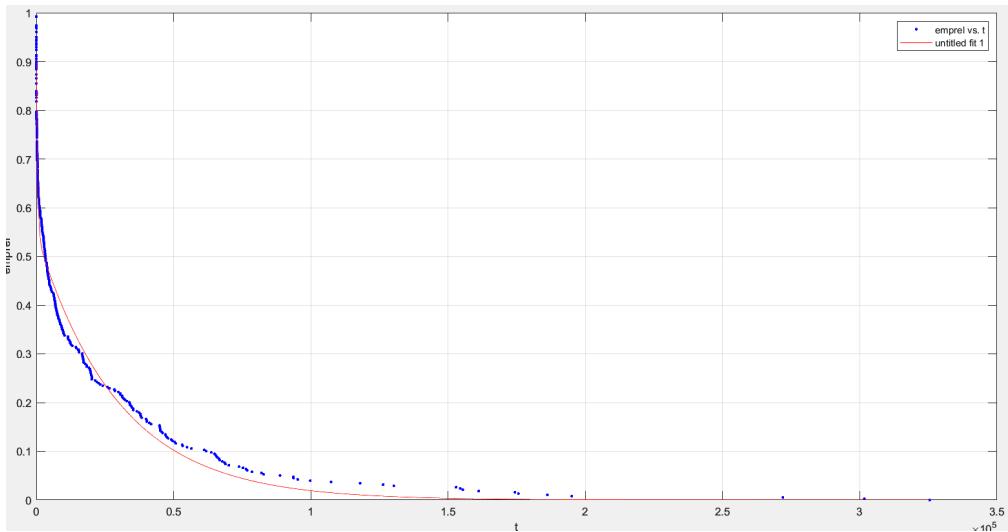


Figura 6.69: Blue Gene IO Card Empirical Reliability Fitting

A valle dell'ottenimento di tali distribuzioni teoriche, si vanno ad effettuare i test per la verifica dell'adattamento. La totalità dei test parametrici effettuati accetta l'ipotesi nulla con un grado di significatività pari a 0.05.

Tipologia	Rel teorica	H-val	p-val	k-val
I-O	Exp2	0	0.5051	0.0634
computational	Exp2	0	0.8135	0.0656

Dal confronto delle reliability empiriche si osserva che il sottosistema di IO ha una reliability peggiore rispetto a quella del sottosistema computazionale, ciò supporta l'ipotesi che il sottosistema di I-O sia un bottleneck. Inoltre la curva di reliability del sistema Blue Gene ha un andamento simile a quella del sottosistema di IO, ciò significa che quest'ultima influenza maggiormente la reliability dell'intero sistema, andando a confermare l'ipotesi di bottleneck.

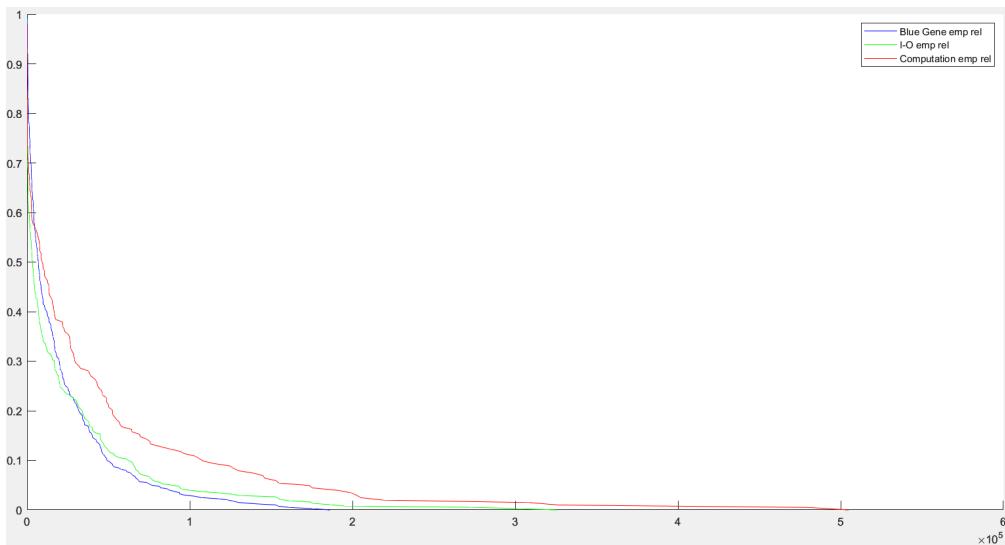


Figura 6.70: Confronto Empirical Reliability

6.4.8 Domande

Si possono usare le stesse finestre di coalescenza in diversi nodi?

Per quanto riguarda i Rack è stato possibile fissare una finestra di coalescenza univoca pari a 75 secondi, per l'analisi delle card computational ed I-O non è stato possibile.

Cosa si può dire della relazione tra l'affidabilità del sistema e l'affidabilità dei nodi?

Si è osservato dalle analisi precedenti che i nodi di I-O risultano avere una reliability molto simile a quella del sistema totale, questo ci suggerisce che siano la tipologia di nodi che influenza maggiormente la reliability totale. Mentre i nodi di tipo computational risultano essere più affidabili.

Esistono colli di bottiglia dell'affidabilità (cioè, i principali contributori al numero totale di fallimenti)?

Come detto in precedenza, un ipotetico collo di bottiglia, è da attribuire all'intero sistema di I-O, visto che la reliability del sistema sembra dipendere fortemente da quella di tali nodi. A differenza di Mercury non è presente un nodo con una reliability peggiore dell'intero sistema.

Nodi funzionali simili (per esempio, due nodi BG/L I/O Ri:Mx:Nz) presentano parametri di affidabilità simili?

6.4.9 Confronto tra Mercury e Blue Gene/L

Dalle analisi precedenti possiamo evidenziare alcune differenze trovate tra i due sistemi. E' possibile affermare che il bottleneck di Mercury è rappresentato dal nodo tg-c401, mentre per il sistema Blue Gene il bottleneck è l'intero sottosistema di IO. La distribuzione temporale degli errori del sistema Mercury presenta un picco in un intervallo di tuple, mentre il sistema Blue Gene presenta errori uniformemente distribuiti nel tempo. Si riportano in figure le reliability empiriche dei due sistemi, dal grafico si evince che il sistema Mercury è meno reliable rispetto a Blue Gene

CAPITOLO 6. FIELD FAILURE DATA ANALYSIS

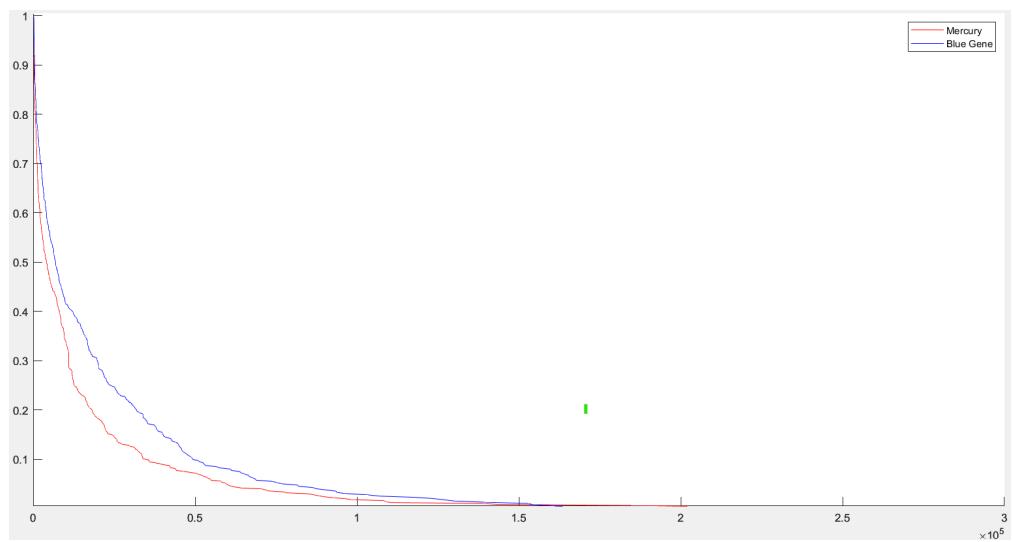


Figura 6.71: Mercury vs Blue Gene Reliability