

# Dependability and Performance Analisys

## esame Impianti di elaborazione

Farina Giorgio - Mat. M63/000861 Giammattei Luca - Mat. M63/000825

13 aprile 2020

# Indice

<b>1 Benchmark</b>	<b>1</b>
1.1 Introduzione . . . . .	1
1.1.1 Progettazione dei test . . . . .	1
1.1.2 Analisi dei risultati . . . . .	2
1.1.2.1 Risoluzione di un problema di dimensione 1000 . . . . .	2
1.1.2.2 Risoluzione di un problema di dimensione 5000, 10000, 15000, 20000 . . . . .	2
1.1.2.3 Risoluzione di un problema di dimensione 1000: nuovi campioni per avere un'accuratezza del 90% . . . . .	3
<b>2 Workload Characterization</b>	<b>5</b>
2.1 Introduzione . . . . .	5
2.1.0.1 Caratterizzazione del Workload e considerazioni iniziali . . . . .	5
2.2 Principal Component Analysis . . . . .	5
2.3 Clustering: ward method . . . . .	6
2.3.1 Calcolo della percentuale di varianza totale spiegata dalla Clusterizzazione . . . . .	7
2.4 Workload sintetico . . . . .	11
<b>3 Capacity Test</b>	<b>13</b>
3.1 Definizione del problema . . . . .	13
3.1.1 Web Client Analysis: . . . . .	13
3.1.2 Web Server Analysis: . . . . .	13
3.2 Workload Characterization . . . . .	14
3.2.1 Simulazione di un workload reale . . . . .	14
3.2.2 Analisi di alto livello . . . . .	14
3.2.3 Web Server Analysis: . . . . .	18
3.3 Capacity Test . . . . .	18
3.3.1 Modo di operare . . . . .	19
3.3.2 Capacity test con richieste di pagine di dimensione 100 KB . . . . .	19
3.3.2.1 Risultati derivati dal modello analitico . . . . .	19
3.3.2.2 Risultati derivati dalla misurazione . . . . .	20
3.3.2.3 Interpretazione dei dati . . . . .	21
3.3.3 Capacity test con richieste di pagine di dimensione 1 MB . . . . .	21
3.3.3.1 Risultati derivati dal modello analitico . . . . .	21
3.3.3.2 Risultati derivati dalla misurazione . . . . .	22
3.3.3.3 Interpretazione dei dati . . . . .	23
3.3.4 Capacity test con richieste di pagine di dimensione 10 MB . . . . .	23

3.3.4.1	Risultati derivati dal modello analitico . . . . .	23
3.3.4.2	Risultati derivati dalla misurazione . . . . .	23
3.3.4.3	Interpretazione dei dati . . . . .	24
3.3.5	Capacity test con richieste di pagine di dimensione random . . . . .	24
3.3.5.1	Risultati derivati dal modello analitico . . . . .	24
3.3.5.2	Risultati derivati dalla misurazione . . . . .	25
3.3.5.3	Interpretazione dei dati . . . . .	26
3.3.6	Conclusione . . . . .	26
3.4	Design of Experiments . . . . .	26
3.4.1	Modello di regressione . . . . .	27
3.4.1.1	Piano degli esperimenti . . . . .	27
3.4.2	Analisi dei risultati . . . . .	27
3.4.2.1	Modello regressivo stimato: . . . . .	27
3.4.2.2	Importanza dei parametri . . . . .	27
3.4.2.3	Ipotesi di normalità: rigettata . . . . .	29
3.4.2.4	Ipotesi di omoschedasticità: rigettata . . . . .	30
3.4.2.5	Significatività dei fattori . . . . .	30
<b>4</b>	<b>Dependability</b>	<b>32</b>
4.1	Esercizio 01 . . . . .	32
4.1.1	Soluzione: conditioning . . . . .	32
4.2	Esercizio 02 . . . . .	33
4.2.1	Soluzione . . . . .	33
4.2.2	Soluzione . . . . .	34
4.3	Esercizio 04 . . . . .	35
4.3.1	Soluzione . . . . .	35
4.4	Esercizio 05 . . . . .	36
4.4.1	Soluzione . . . . .	38
4.4.1.1	Quesito 1 . . . . .	38
4.4.1.2	Quesito 2 . . . . .	38
4.4.1.3	Quesito 3 . . . . .	39
4.4.1.4	Quesito 4 . . . . .	40
<b>5</b>	<b>FFDA</b>	<b>41</b>
5.1	Introduzione . . . . .	41
5.2	FFDA: Mercury . . . . .	41
5.2.1	Traccia . . . . .	41
5.2.2	Soluzione . . . . .	42
5.2.2.1	Quesito 1 . . . . .	42
5.2.2.2	Quesito 2 . . . . .	45
5.2.2.3	Quesito 3 . . . . .	48
5.2.2.4	Quesito 4 . . . . .	52
5.2.2.5	Quesito 5 . . . . .	53
5.3	FFDA: Blue Gene . . . . .	53
5.3.1	Traccia . . . . .	54
5.3.2	Soluzione . . . . .	55

5.3.2.1	Quesito 1 . . . . .	55
5.3.2.2	Quesito 2 . . . . .	57
5.3.2.3	Quesito 3 . . . . .	60
5.3.2.4	Quesito 4 . . . . .	62
5.3.2.5	Quesito 5 . . . . .	63
<b>6</b>	<b>Cloud</b>	<b>64</b>
6.1	Cloud . . . . .	64
6.1.1	Traccia . . . . .	64
6.1.2	quesito 1 . . . . .	64
6.1.3	quesito 2 . . . . .	65

# Capitolo 1

## Benchmark

### 1.1 Introduzione

L'obiettivo di questa analisi di performance è comparare le prestazioni di due sistemi nell'eseguire operazioni floating point. Se i due sistemi sono significativamente differenti con un livello di confidenza del 90 % e un'accuratezza del 90%.

Nello specifico si utilizzerà un benchmark della Intel, Linpack, che consiste nella risoluzione di sistemi di equazioni lineari densi in cui i coefficienti sono generati randomicamente.

I due sistemi sono:

Dell XPS13, OS windows 10, Intel i7-7560U @ 2.40GHz (4 MB di cache), RAM 16 GB

Macbook Pro, MacOS High Sierra, Intel i7-7820HQ @ 2.90 GHz (8 MB di cache), RAM 16 GB

Le metriche utilizzate sono MFLOPS e il response time.

#### 1.1.1 Progettazione dei test

Di seguito in Tab. è mostrato come sono settati i parametri del benchmark.

Poichè sono stati condotti n esperimenti su ciascuno dei due sistemi così che l'i-esimo test sul sistema A e l'i-esimo test sul sistema B hanno una corrispondenza one-to.one, allora tali osservazioni sono paired.

I due campioni è possibile trattarli come un unico campione di n pairs. Per ciascun pair è stata calcolata la differenza e per la differenza verrà valutato l'intervallo di confidenza.

Se l'intervallo di confidenza includerà lo zero, allora i sistemi non sono significativamente differenti, ovvero le varie osservazioni sono realizzazioni della stessa popolazione, e le differenze sono dovute semplicemente al caso.

Per il dimensionamento del campione si è scelto di campionare inizialmente 10 osservazioni per ciascun test, se tale numero non risulta adatto per ottenere il livello di confidenza e accuratezza desiderato sarà necessario campionare nuove osservazioni.

Dimensione della matrice	1000	5000	10000	15000	20000	10 ripetizioni
Dimensione della matrice	1000	5000	10000	15000	20000	10 ripetizioni

Tabella 1.1: Settaggio dei parametri di LINPACK

## 1.1.2 Analisi dei risultati

### 1.1.2.1 Risoluzione di un problema di dimensione 1000

Solo con 10 osservazioni è stato possibile dimostrare con un livello di confidenza del 90% e un'accuratezza del 94% che le prestazioni del MAC in termini di response time è significativamente più performante di quella del DELL.

Trenta osservazioni di ritiene essere un numero sufficientemente alto affinchè la distribuzione delle medie campionarie possa essere letta secondo una legge normale, avendo qui utilizzato un campione di appena 10 osservazioni, la distribuzione delle medie campionare è stata approssimata a una t-student di grado 9.

DELL	0.013	0.012	0.011	0.010	0.010	0.010	0.010	0.010	0.010	0.010							
MAC	0.005	0.005	0.005	0.006	0.005	0.006	0.005	0.005	0.005	0.005							
Differenze	0.008	0.007	0.006	0.004	0.005	0.004	0.005	0.005	0.005	0.005							
	0,000 0067 6	0.000 0025 6	0.000 0003 6	0.00000196	0.000 0001 6	0.000 0019 6	0.000 0001 6	0.000 0001 6	0.000 0001 6	0.000 0001 6							
Media delle differenze	0.0054			dev.stand			0.0012649										
intervallo di confidenza	$0.0054 \pm t[95; 9] * 0.0013/\sqrt{10} = 0.0054 \pm 1.833 * 0.0013/\sqrt{10} = 0.0054 \pm 0.00075$																
Accuratezza	86%																

Tabella 1.2: Response time

Media delle differenze	60,8506	deviazione standard	9.4046
intervallo di confidenza	$60.8506 \pm t[95; 9] * 9.4046/\sqrt{10} = 60.8506 \pm 1.833 * 9.4046/\sqrt{10} = 60.8506 \pm 5.45$		
Accuratezza	91%		

Tabella 1.3: MFLOPS

### 1.1.2.2 Risoluzione di un problema di dimensione 5000, 10000, 15000, 20000

Per gli analoghi calcoli per le altre dimensioni del problema si è ricorso a uno script in matlab, in fig. sono mostrati gli output di medie, intervalli di confidenza, e accuratezza sia per il response time che per MFLOPS

Per tutti è stato dimostrato che gli intervalli di confidenza non comprendono lo zero, quindi le prestazioni del MAC sia in termini di response time che di throughtput sono significativamente maggiori del DELL, inoltre si ha un'accuratuzza maggiore del 90 % per tutti e quattro i risultati.

<b>RT_DELL = 4x10</b>
1.1440 1.0780 1.0220 1.0140 1.0300 1.0490 1.0580 1.0770 1.0940 1.1120
8.8140 9.4020 9.7260 9.7990 9.7160 9.6870 9.7010 9.6570 10.3350 9.6280
32.4380 31.9840 31.9680 31.9210 34.8980 31.9730 31.9560 32.2220 33.0040 31.9290
75.7620 74.4800 74.7750 75.6040 74.3730 74.1560 75.1140 81.7610 74.2700 76.6590
<b>TP_DELL = 4x10</b>
72.8912 77.3597 81.5828 82.2207 80.9223 79.4995 78.7920 77.4466 76.2028 74.9597
75.6588 70.9297 68.5642 68.0552 68.6331 68.8406 68.7439 69.0570 64.5259 69.2649
69.3766 70.3608 70.3966 70.5015 64.4859 70.3856 70.4227 69.8418 68.1864 70.4831
70.4061 71.6180 71.3353 70.5537 71.7218 71.9317 71.0135 65.2403 71.8211 69.5831
<b>RT_MAC = 4x10</b>
0.5600 0.5280 0.5270 0.5430 0.5260 0.5250 0.5280 0.5760 0.5250 0.5250
3.9160 3.9750 4.0280 4.0050 4.1060 4.1140 4.1130 4.1370 4.1010 4.1520
13.8570 14.0300 13.4660 13.6320 18.2560 13.5460 14.1470 13.5690 13.7400 13.7070
32.3930 32.2140 32.1290 32.2110 32.2500 32.3470 32.7310 32.1070 32.3470 32.4520
<b>TP_MAC = 4x10</b>
148.7787 158.0173 158.1234 153.5166 158.4551 158.6948 157.7760 144.6480 158.7519 158.9410
170.2831 167.7789 165.5635 166.5260 162.4133 162.0987 162.1488 161.2146 162.6298 160.6195
162.4032 160.3983 167.1203 165.0849 123.2720 166.1333 159.0740 165.8506 163.7900 164.1793
164.6718 165.5855 166.0248 165.6013 165.4013 164.9042 162.9688 166.1361 164.9030 164.3697
<b>Diff_RT = 4x10</b>
0.5840 0.5500 0.4950 0.4710 0.5040 0.5240 0.5300 0.5010 0.5690 0.5870
4.8980 5.4270 5.6980 5.7940 5.6100 5.5730 5.5880 5.5200 6.2340 5.4760
18.5810 17.9540 18.5020 18.2890 16.6420 18.4270 17.8090 18.6530 19.2640 18.2220
43.3690 42.2660 42.6460 43.3930 42.1230 41.8090 42.3830 49.6540 41.9230 44.2070
<b>medie_RT = 1x4</b>
0.5315 5.5818 18.2343 43.3773
<b>devStandard_RT = 1x4</b>
0.0399 0.3319 0.6882 2.3331
<b>IntConfidenza_RT = 1x4</b>
0.0231 0.1924 0.3989 1.3524
<b>Accuratezza_RT = 1x4</b>
0.9565 0.9655 0.9781 0.9688
<b>Diff_TP = 4x10</b>
75.8875 80.6576 76.5406 71.2959 77.5328 79.1953 78.9840 67.2014 82.5491 83.9813
94.6243 96.8492 96.9993 98.4708 93.7802 93.2581 93.4049 92.1576 98.1039 91.3546
93.0266 90.0375 96.7237 94.5834 58.7861 95.7477 88.6513 96.0088 95.6036 93.6962
94.2657 93.9675 94.6895 95.0476 93.6795 92.9725 91.9553 100.8958 93.0819 94.7866
<b>medie_TP = 1x4</b>
77.3826 94.9003 90.2865 94.5342
<b>devStandard_TP = 1x4</b>
5.0606 2.5292 11.3768 2.4300
<b>IntConfidenza_TP = 1x4</b>
2.9333 1.4661 6.5945 1.4085
<b>Accuratezza_TP = 1x4</b>
0.9621 0.9846 0.9270 0.9851

Figura 1.1: Medie, intervalli di confidenza, e accuratezza sia per il response time che per il throughput

### 1.1.2.3 Risoluzione di un problema di dimensione 1000: nuovi campioni per avere un'accuratezza del 90%

L'accuratezza del response time è dell'86%, volendo avere invece una accuratezza del 90% è doveroso rifare il test scegliendo un numero di campioni più alto.

Per ottenere una stima del numero di campioni necessario ad ottenere tle accuratezza si suppone che la varianza e la media rimangano le stesse anche per un numero maggiore di campioni.

Bisogna però considerare anche la variazione del quantile della t-student che varia rispetto alla dimensione del campione.

$$\text{Accuratezza} = \frac{\frac{\text{dev.standard}}{\sqrt{N}} t_{[95, N-1]}}{\text{Media}} \text{ allora } N = \left( \frac{\text{dev.Standard} * t_{[95, N-1]}}{\text{Media} * 0,05} \right)^2 = \left( \frac{0,0012649 * 1,833}{0,0054 * 0,05} \right)^2 = 74$$

```
medie_RT = 0.0045
devStandard_RT = 0.0017

IntConfidenza_RT = 3.3384e-04
Accuratezza_RT = 0.9256

medie_TP = 57.1472
devStandard_TP = 11.9071

IntConfidenza_TP = 2.2770
Accuratezza_TP = 0.9602
```

Figura 1.2: Risultati ottenuti con un campione di 74 elementi per la risoluzione di un problema di dimensione 1000

I risultati sono mostrati in fig. . Le performance del MAC risultano ancora essere significativamente superiori con un livello di confidenza del 90 %, inoltre è stato raggiunto per il tempo di risposta un'accuratezza superiore al 90 %.

In realtà come quantile dello 0.95 è stato utilizzato non quello della t-student, ma quello della normale essendo il campione con dimensione notevolmente maggiore di 30.



# Capitolo 2

## Workload Characterization

### 2.1 Introduzione

Analizzare i dati contenuti all'interno del file EsercizioPCA\_Clustering\_versStudenti.xlsx , che rappresentano un workload reale ottenuto da un sistema operativo linux-based. L'obiettivo dell'e-laborato è ottenere un workload sintetico che sia rappresentativo di quello reale, preservando gran parte della varianza.

#### 2.1.0.1 Caratterizzazione del Workload e considerazioni iniziali

Come primo passo nella caratterizzazione del workload reale, si è pensato di caratterizzare ogni parametro con la media aritmetica dei suoi valori osservati. Utilizzando come indice di dispersione il coefficiente di variazione, è stato osservato che esso (ottenuto come il rapporto tra la deviazione standard e la media) è pari a zero per quattro parametri. Questo significa che le osservazioni per quei parametri risultano costanti e quindi possono essere rappresentati dalla loro media. I parametri sono “avgLatency”, “Errors”, “AnonPages”, “Active”. Il parametro “Slab” ha la particolarità che tutti i percentili fino al 99.5% sono pari a zero e il massimo risulta pari a ‘1760’. Quest'ultimo è assunto da una sola osservazione, identificata come Outlier, la quale sarà analizzata individualmente per decidere se includerla o no nel modello. Il coefficiente di variazione risulta grande per tutti gli altri parametri quindi questi ultimi non possono essere rappresentati in modo significativo dall'indice di tendenza centrale. A questo punto si potrebbe pensare di proseguire sintetizzando i parametri (fattori) con la loro distribuzione, ma questa opzione non è stata considerata in quanto i fattori risultano correlati come ci indica sia la matrice di correlazione (coefficienti di correlazione in valore assoluto vicini a uno) sia l'applicazione della PCA da cui risulta che sole 9 componenti su 19 sono in grado di spiegare il 99,315% della varianza. La correlazione comporta che certe combinazioni di valori dei parametri non rappresentano il comportamento del workload reale, e quindi non avrebbe senso considerarle.

### 2.2 Principal Component Analysis

La PCA è una trasformazione lineare  $Y = PX$  del campione  $X$  (Workload reale normalizzato a media nulla e varianza unitaria), dove si impone che la matrice di covarianza  $C_Y$  di  $Y$  sia una matrice diagonale. Si dimostra che  $C_Y$  coincide con la matrice diagonale degli autovalori della

matrice di covarianza di  $X$  e quindi ogni autovalore corrisponde alla varianza spiegata da una componente principale. In fig.?? sono riportati gli autovalori con la percentuale di varianza totale spiegata.

Inoltre si dimostra che la matrice di trasformazione  $P$  è la matrice degli autovettori della matrice di covarianza di  $X$ . E' una matrice ortonormale grazie alla simmetria della matrice di Covarianza di  $X$ . Le  $Y_i$  sono ortogonali e di conseguenza indipendenti. In fig.?? è riportato il loading plot ( $X = P^T Y$ ) dove l'elevata correlazione tra due variabili  $X_i$  comporta che i due vettori siano molto vicini tra loro, la non correlazione comporta che i due vettori abbiano uno sfasamento di  $90^\circ$  ed infine la correlazione negativa uno sfasamento di  $180^\circ$ . I vettori in figura sono dotati di sole due componenti, le due più significative risultanti dalla PCA. La proiezione del vettore su ogni componente principale è la rappresentazione del peso che quella componente ha sul parametro. Si osservi infatti che i vettori sono tendenzialmente più vicini all'asse delle ascisse ovvero alla prima componente principale.

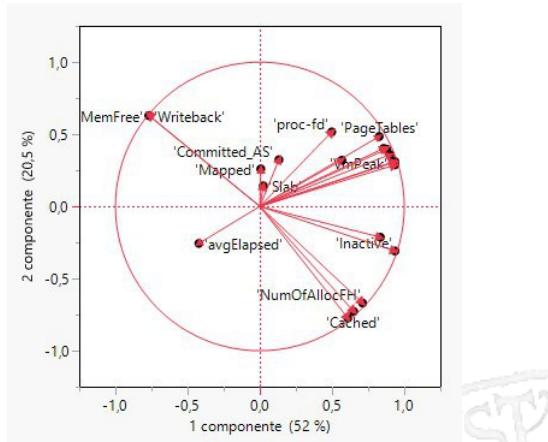


Figura 2.1: Autovalori con percentuale cumulativa

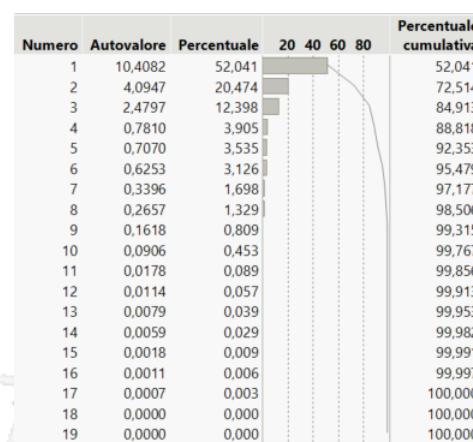


Figura 2.2: Autovalori con percentuale cumulativa

Dall'applicazione della PCA risulta che con sole 9 componenti principali su 19 sono in grado di esprimere il 99,315% della varianza totale. Per avere una percentuale superiore al 90% tuttavia basterebbero soltanto 5 componenti. Si è deciso comunque di prenderne 9 così da tenere in conto anche la perdita dovuta al successivo passo di clustering. Inoltre l'applicazione del Clustering sul nuovo insieme campione trasformato  $Y$  piuttosto che sul campione di partenza  $X$  risulta più consistente grazie all'incorrelazione che sussiste tra le componenti principali.

## 2.3 Clustering: ward method

L'obiettivo è di ottenere un workload sintetico che abbia ridotto di molto il numero di componenti, ma che continui a spiegare il 90% della varianza del workload reale. Con l'applicazione della PCA è stato ottenuto un workload "trasformato"  $Y$ , del quale si è deciso di considerare solo 9 componenti principali spiegando il 99,315% del workload reale. Le nuove osservazioni corrispondenti alle componenti principali (i vettori  $Y_i$ ) risultano ortogonali, e quindi indipendenti. Questa condizione è significativa per avere fiducia nei risultati ottenuti dall'applicazione del Clustering. E' stato appli-

cato come metodo di clusterizzazione quello di ward. E' un metodo di clusterizzazione gerarchico, single linkage, che considera i centroidi come punti rappresentativi dei cluster, e che come metrica di distanza utilizza la distanza di ward.

Il workload ridotto dalla clusterizzazione deve spiegare almeno il 91% della varianza del campione (con nove componenti principali) sintetizzato dalla PCA, in questo modo il workload sintetico risultante spiegherà almeno il 90% del workload reale. Il problema si riconduce nello scegliere in modo opportuno il numero di cluster, affinchè la percentuale di varianza intercluster spiegata sia del 91%. Per individuare il numero di cluster è stata calcolata la percentuale di varianza intercluster spiegata per un numero di cluster che varia da uno a 20.

### 2.3.1 Calcolo della percentuale di varianza totale spiegata dalla Clusterizzazione

La funzione utilizzata nel calcolare la percentuale di varianza intercluster è riportata in Alg.2.2. Esso si basa sulla seguente spiegazione matematica.

Essendo le osservazioni lungo le componenti principali (i vettori  $Y_i$ ) ortogonali, la devianza totale può essere calcolata come la somma delle devianze delle 9 componenti principali  $SS1 + SS2 + \dots + SS9 = SST$ , da cui

$$\frac{SS1}{SST} + \frac{SS2}{SST} + \dots + \frac{SS9}{SST} = 1$$

La percentuale di varianza spiegata dal workload costituito dai centroidi dei cluster dopo la clusterizzazione quindi sarà

$$\frac{SSClusterC1}{SST} + \frac{SSClusterC2}{SST} + \dots + \frac{SSClusterC9}{SST} < 1$$



Figura 2.3: Porzione della variazione totale di ciascuna colonna assorbita dalla clusterizzazione (13 cluster)

Da JMP è possibile ottenere, per un certo numero di cluster, gli R-quadro , ovvero la porzione di varianza totale di ciascuna colonna assorbita dalla clusterizzazione ( $\frac{SSClusterC_i}{SSi} = R_i^2$  ).

In fig.2.3 sono riportati gli R-quadro con tredici cluster; ad esempio la percentuale di varianza totale della componente principale 1 assorbita dalla clusterizzazione è  $0.9754 = SSSClusterC1/SS1$

Generalizzando  $\frac{SSClusterC_i}{SSi} = R_i^2$  , da cui  $SSClusterC_i = R_i^2 * SSi$

Quindi selezionando come numero di cluster 13 la percentuale di varianza totale spiegata dal workload sintetico ottenuto dalla clusterizzazione è pari a

$$0.9754 * \left(\frac{SS1}{SST}\right) + 0.9789 * \left(\frac{SS2}{SST}\right) + \dots + 0.8883 * \left(\frac{SS9}{SST}\right) = 0.91$$

### Algoritmo 2.1 Calcolo dei pesi

```

1 DevColPCA=zeros(9,1);
2 TotDevianza1=0;
3 m= 3000;
4 for i = 1:1:9
5
6     for j=1:1:m
7         DevColPCA(i) = DevColPCA(i)+(pca(j,i)-medTot(i))^2;
8     end
9     TotDevianza1= TotDevianza1 + DevColPCA(i);
10 end
11
12
13 DevColPCAPesi = DevColPCA./TotDevianza1;
```

**Algoritmo 2.2** Calcolo percentuale di varianza spiegata dalla clusterizzazione tramite l'utilizzo di  $R^2$

```

1 function [PercInterCluster,PercIntraCluster] = calcoloPercVarInterIntra(
2     DevColPCAPesi,PCARSquare)
3 PercInterCluster = DevColPCAPesi'*PCARSquare;
4 PercIntraCluster = 1-PercInterCluster;
5 end

```

Per avere una conferma di quanto calcolato è stato implementato anche uno script che calcolasse la devianza intercluster, intracluster e totale. I risultato sono stati calcolati per un numero di cluster pari a 10 per poi essere confrontati con quelli ottenuti dal calcolo tramite l'alg.2.2 sfruttando  $R^2$ .

```

n = 10;
m = 3000;
Intracluster=zeros(n,1);
TotIntraCluster=0;
for i = 1:1:n
    if isnan(deviazioni10cl(i,3))~=1
        for j=3:1:11
            Intracluster(i) = Intracluster(i)+(deviazioni10cl(i,j)^2*(deviazioni10cl(i,2)-1));
        end
    end
    TotIntraCluster= TotIntraCluster + Intracluster(i);
end

Intercluster=zeros(n,1);
TotInterCluster=0;
for i = 1:1:n

    for j=3:1:11
        Intercluster(i) = Intercluster(i)+(medie10cl1(i,j)- medTot(j-2))^2;
    end

    TotInterCluster= TotInterCluster + medie10cl1(i,2)*Intercluster(i);
end

DevComponenti=zeros(m,1);
TotDevianza=0;
for i = 1:1:m

    for j=1:1:9
        DevComponenti(i) = DevComponenti(i)+(pca(i,j) - medTot(j))^2;
    end

    TotDevianza= TotDevianza + DevComponenti(i);
end

TotDevianza
TotInterCluster
TotIntraCluster
PercentualeVarianza = TotInterCluster/TotDevianza
PercentualeVarianzaPersa = TotIntraCluster/TotDevianza

```

```

TotDevianza = 5.9569e+04
TotInterCluster = 4.8871e+04
TotIntraCluster = 1.0698e+04
PercentualeVarianza = 0.8204
PercentualeVarianzaPersa = 0.1796

```

Figura 2.4: Calcolo devianza Inter Cluster, Intra Cluster e Totale

Figura 2.5: Funzione che calcola la percentuale di varianza spiegata avendo come input Rsquare; applicazione di tale funzione al variare del numero di cluster

La percentuale di varianza totale assorbita dalla clusterizzazione è stata calcolata al variare del numero di cluster tramite i valori di  $R^2$  forniti da JMP e per mezzo dell'alg.2.2. Le percentuali calcolate sono mostrate numericamente nelle fig.?? e ?? e graficamente nelle fig.?? e ??.

Si osservi come da una clusterizzazione di 11 cluster a una di 12 vi sia un forte aumento di percentuale di varianza spiegata. Per avere una percentuale spiegata almeno del 91% è sufficiente avere tredici cluster.

1	0
2	0.0740
3	0.3981
4	0.4542
5	0.6602
6	0.6926
7	0.7520
8	0.7661
9	0.8011
10	0.8204
11	0.8370
12	0.9025
13	0.9162
14	0.9230
15	0.9283
16	0.9368
17	0.9476
18	0.9496
19	0.9513
20	0.9530
21	0.9546
22	0.9633
23	0.9646
24	0.9652
25	0.9663
30	0.9705
40	0.9793
50	0.9839
60	0.9870
70	0.9885
80	0.9895
90	0.9911
100	0.9922

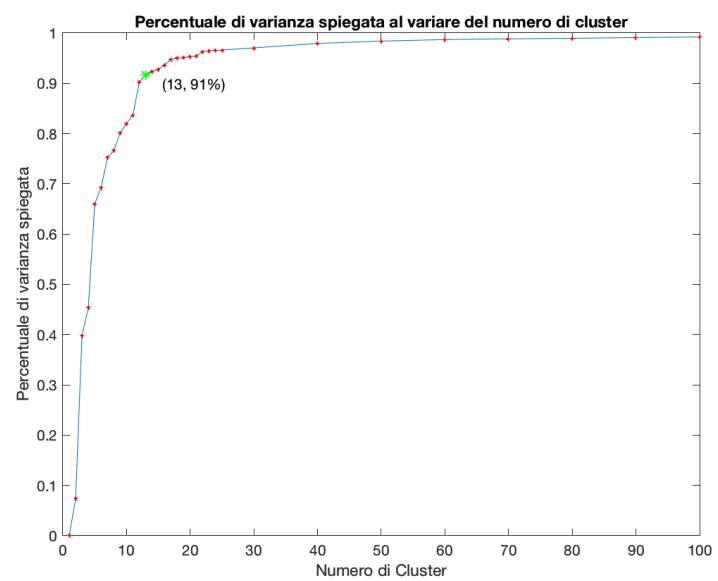


Figura 2.6: A sinistra il numero di cluster a destra la percentuale di varianza totale conservata

Figura 2.7: Percentuale di varianza totale assorbita dalla clusterizzazione al variare del numero di cluster

1	1
2	0.9260
3	0.6019
4	0.5458
5	0.3398
6	0.3074
7	0.2480
8	0.2339
9	0.1989
10	0.1796
11	0.1630
12	0.0975
13	0.0838
14	0.0770
15	0.0717
16	0.0632
17	0.0524
18	0.0504
19	0.0487
20	0.0470
21	0.0454
22	0.0367
23	0.0354
24	0.0348
25	0.0337
30	0.0295
40	0.0207
50	0.0161
60	0.0130
70	0.0115
80	0.0105
90	0.0089
100	0.0078

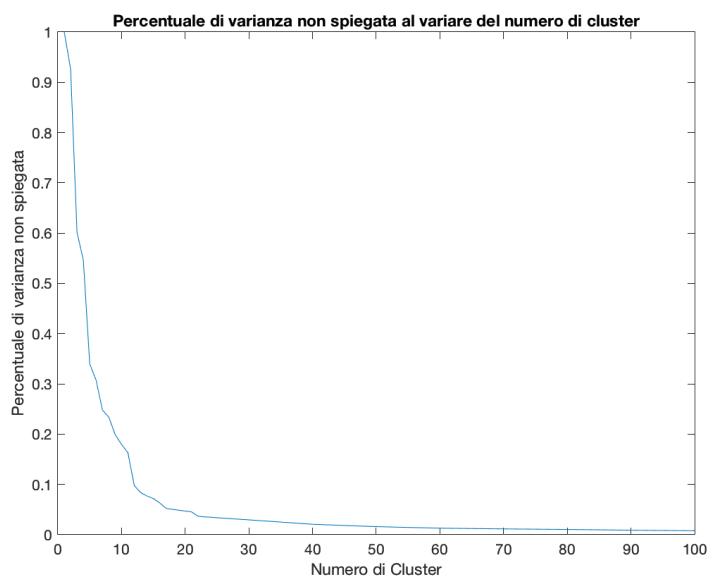


Figura 2.9: Percentuale di varianza totale assorbita dalla clusterizzazione al variare del

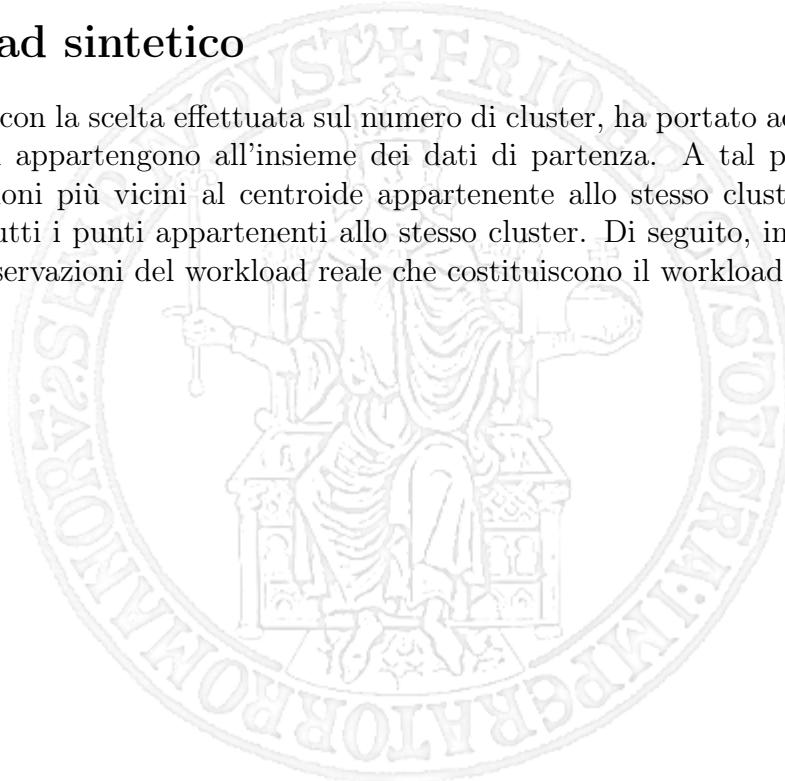
Figura 2.8: A sinistra il numero di cluster numero di cluster

a destra la percentuale di varianza totale

conservata

## 2.4 Workload sintetico

La clusterizzazione, con la scelta effettuata sul numero di cluster, ha portato ad avere 13 centroidi. Tuttavia questi non appartengono all'insieme dei dati di partenza. A tal proposito si è deciso di prendere i campioni più vicini al centroide appartenente allo stesso cluster. Questi saranno rappresentativi di tutti i punti appartenenti allo stesso cluster. Di seguito, in fig., sono riportate le corrispondenti osservazioni del workload reale che costituiscono il workload sintetico.



## CAPITOLO 2. WORKLOAD CHARACTERIZATION

'VmPeak'	'VmSize'	'VmHWM'	'VmRSS'	'VmPTE'	'Threads'	'MemFree'	'Buffers'	'Cached'	'Active'	'Inactive'	'Dirty'
152272	150216	25228	25208	160	34	5604888	33144	334652	0	141436	303848
153484	151436	26612	26468	160	32	5084564	83540	693480	0	327380	575804
170344	170340	31208	31140	200	13	4709868	97944	1075684	0	376648	882056
170344	170340	31144	31140	200	43	4471580	94884	1259568	0	597900	891332
172388	170340	32696	31456	200	33	4638372	129452	1117180	0	411940	918492
189652	184388	41320	38024	252	96	4559008	196300	1118044	0	483172	921536
172392	170340	32868	31340	200	28	4592848	173256	1117624	0	441136	933428
192256	187216	43804	39716	264	87	4553128	199380	1118164	0	493172	916428
176420	174376	34220	33824	208	35	4570732	191052	1117916	0	468576	926556
192256	185444	43804	37600	252	100	4556264	199112	1118152	0	489952	917252
209208	206136	56080	54712	308	161	4534584	203968	1118384	0	520396	908972
215352	206136	61784	53204	312	9	4539344	204628	1118492	0	521100	907540
170344	170340	31144	31140	200	52	4655664	89480	1090572	0	541220	773332

'Writeback'	'AnonPage'	'Mapped'	'Slab'	'PageTable'	'Committe'	'NumOfAll'	'proc-fd'	'avgThroug'	'avgElapse'	'avgLatenc'	'Errors'
5604888	8159224	36	0	77472	23772	27832	7196	296468	1530	0	2
5084564	8159224	121820	0	126116	28324	90284	7984	343200	2040	0	2
4709868	8159224	119972	0	85176	24272	109568	7704	317992	1530	0	2
4471580	8159224	459632	0	134504	29956	116216	8024	365692	1530	0	2
4638372	8159224	168	0	83800	23740	109472	7208	319468	1530	0	2
4559008	8159224	60	0	90300	23848	113948	7256	329996	1020	0	2
4592848	8159224	144	0	83684	23844	110612	7204	315912	1020	0	2
4553128	8159224	40	0	92252	23852	114044	7476	334536	1020	0	2
4570732	8159224	12	0	86168	23844	112628	7212	321656	1530	0	2
4556264	8159224	172	0	90184	23848	113856	7036	332608	1020	0	2
4534584	8159224	36	0	107040	23848	112708	7312	349376	1020	0	2
4539344	8159224	136	0	105548	23848	110200	7320	349868	1020	0	2
4655664	8159224	324400	1760	134500	29956	106788	8028	365264	2040	0	2

Figura 2.10: Workload sintetico



# Capitolo 3

## Capacity Test

### 3.1 Definizione del problema

Si è creato un sistema Client-Server che comunica tramite una rete privata in cui è richiesto che il tempo di risposta sia inferiore in media a un secondo indipendentemente dal tipo di richiesta http. Si è ipotizzato, inoltre, che gli utenti inviano una richiesta ogni 250 ms (un quarto di secondo) e possano richiedere tre tipi di pagine, ognuna di grandezza differente (100KB, 1MB, 10MB).

L'analisi del sistema è stata divisa secondo una prospettiva web client e una prospettiva server.

#### 3.1.1 Web Client Analysis:

Ci si è posti nella prospettiva di un utente, intesa come persona, che tramite l'interfaccia (ad esempio un browser), voglia aprire una pagina web.

Le condizioni di ambiente suppongono che il tempo di latenza della rete sia trascurabile e che il tempo di visualizzazione e interpretazione della pagina da parte del browser sia trascurabile.

L'obiettivo è valutare se il tempo di risposta alle richieste del client è legato in qualche modo al tipo di richiesta effettuata dal client.

Servizi: Visitare una pagina

Fattori: Page size

Metriche: Response time (elapsed time e latency)

Workload: Lista di pagine

#### 3.1.2 Web Server Analysis:

Il sistema sotto studio è il server: il servizio offerto dal Server è quello di accettare connessioni e inviare delle pagine html.

L'obiettivo è valutare la produttività di servizio del Server considerando come fattori aggiuntivi i parametri che descrivono lo stato corrente del sistema su cui il server è in esecuzione.

Servizi: Accettare e validare connessioni, e Invio di pagine

Fattori: CPU, System, procs, io, memory and swap parameters, dimensione della pagina da trasferire(piccole, medie, grandi)

Metrica: Throughput

Workload: Richieste HTTP

## 3.2 Workload Characterization

Gli obiettivi che si andranno ad affrontare in questo paragrafo sono:

- Generare un random workload
- Collezionare parametri del workload ad application-level
- Collezionare parametri del workload a system-level
- Analizzare con diverse tecniche i dati raccolti

### 3.2.1 Simulazione di un workload reale

Per simulare una situazione tipica del problema Client Server sono state fatte alcune considerazioni.

Inizialmente è stato stimato il tempo di servizio come il tempo impiegato mediamente dal server nell'eseguire una sola richiesta che è risultato essere di 12.75 m.

Considerando le specifiche del problema il tempo di think time medio che intercorre tra la risposta ed una successiva richiesta del client è di 250 ms.

I risultati derivati da una modellazione matematica del problema sono:

- un carico di utenti ottimo pari a 20 ( $N^* = 1 + \text{TempoRichiestaMedio}/\text{TempoServizioMedio}$ )
- per un numero di utenti pari a 40 si ha un tempo di risposta limitato inferiormente da 260 ms. ( $E[T] = N^*\text{TempoServizioMedio} - \text{TempoRichiestaMedio}$ )

Il carico di utenti è stato scelto pari a 40 in quanto, supponendo che il tempo di risposta in 40 sia il doppio del limite inferiore (500 ms), esso risulta essere ancora un tempo di risposta accettabile secondo le specifiche del problema.

### 3.2.2 Analisi di alto livello

Dalla generazione del workload sono stati collezionati diversi dati, utili ad una breve ma importante analisi. In particolare i dati più significativi presi in considerazione sono: timestamp, elapsed, latency, bytes.

La variabile Bytes rappresenta i byte ricevuti in seguito ad una richiesta e infatti assume soltanto 3 valori sull'intero workload, “123943” per richieste da 100KB, “1056371” per richieste da 1MB ed infine “10441491” per quelle da 10 MB.

In primo luogo si è deciso di calcolare la matrice di correlazione e approfondire il legame tra i valori dei tempi di risposta elapsed e latency. Come si può notare dalla fig.3.1, le correlazioni tra le variabili sono quasi tutte basse a meno della correlazione tra bytes e elapsed che, come ci si immaginava, a differenza di quella con Latency risulta molto significativa.

	timeStamp	bytes	Latency	elapsed
timeStamp	1,0000	0,0126	0,1897	0,1022
bytes	0,0126	1,0000	0,0063	0,6661
Latency	0,1897	0,0063	1,0000	0,1153
elapsed	0,1022	0,6661	0,1153	1,0000

Figura 3.1: Matrice di correlazione

A prova di quanto affermato e osservato dalla matrice delle correlazioni, possiamo osservare le successive figure, dove troviamo rappresentati i grafici timestamp-elapsed e timestamp-latency, con la sovrapposizione dei bytes. Osserviamo che per la variabile elapsed c'è una forte dipendenza dal tipo di richiesta effettuata, al contrario di Latency, dove tutte le rette risultanti dai dati si sovrappongono.

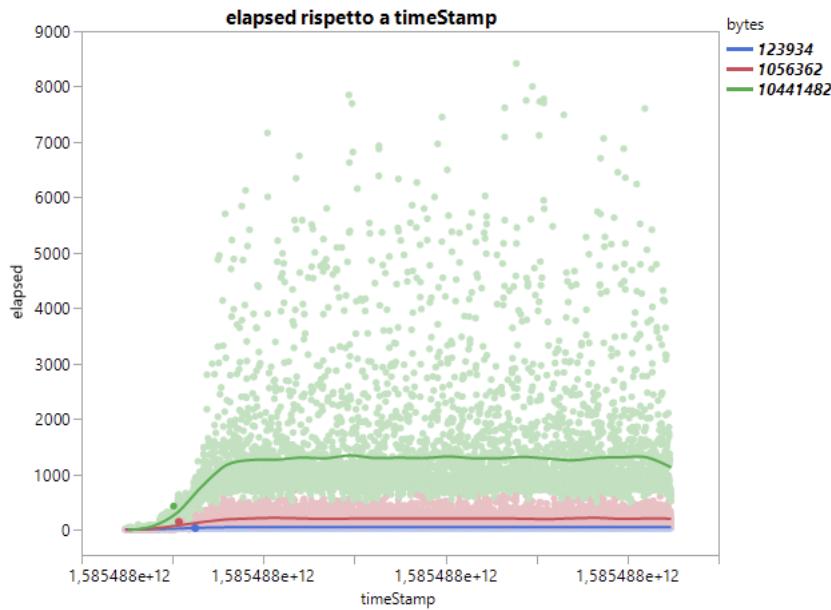


Figura 3.2: Evoluzione nel tempo dell'Elapsed time per i tre tipi di pagina

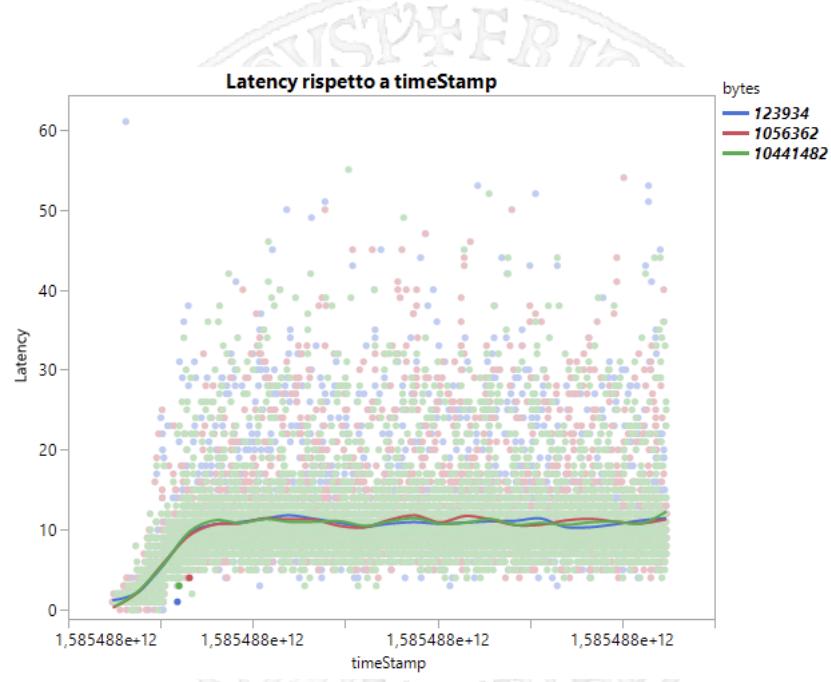


Figura 3.3: Evoluzione nel tempo della Latency per i tre tipi di pagina

Analisi di basso livello

Tramite il comando vmstat è stato possibile raccogliere alcune informazioni relative alla macchina virtuale lato server.

In particolare i parametri sono:

- r: numero di processi pronti;
- b: numero di processi in sleep;
- swpd: quantità di memoria virtuale utilizzata
- free: quantità di memoria inutilizzata;
- inact: quantità di memoria inattiva;
- active: quantità di memoria attiva;
- si: Fattore di ritorno dello swap (da disco a RAM);
- so: Fattore di uscita dello swap (da RAM a disco);
- bi: numero di blocchi al secondo ricevuti dal disco;
- bo numero di blocchi al secondo inviati al disco;
- in: interrupts al secondo;
- cs: context switch al secondo;
- us: percentuale di tempo in cui la CPU ha eseguito codice utente;
- sy: percentuale di tempo in cui la CPU ha eseguito codice kernel;
- id: percentuale di tempo in cui la CPU è in idle;
- wa: percentuale di tempo in cui la CPU è in attesa di operazioni di I/O;
- st: percentuale di tempo rubata dalla macchina virtuale.

Si è deciso di effettuare una analisi dei parametri preliminare, osservando le distribuzioni e gli indici di tendenza centrale e di dispersione, oltre che una analisi della matrice di correlazione.

Le considerazioni effettuate sono state:

alcune variabili, essendo costanti, non portano variabilità e per questo motivo è possibile eliminarle (swpd, si, st)

- I parametri ‘so’ e ‘b’ presentano soltanto due valori diversi da 0, in corrispondenza dell’inizio della registrazione dei dati, quando il test lato client non era ancora iniziato, per cui è possibile eliminarli
- Le restanti variabili presentano variabilità e correlazioni non trascurabili

- Dalla matrice di correlazione si può evincere che, come si poteva immaginare, vi è una forte correlazione tra ‘cs’ (context-switch al secondo) con ‘in’ (numero di interruzioni) oltre che con ‘sy’ (tempo CPU per codice kernel) e ‘active’ (quantità di memoria attiva). In generale si può notare che sono presenti correlazioni negative tra tutte le variabili per le quali un valore alto in una rappresenta uno “scarso” utilizzo del sistema mentre nell’altra rappresenta un “forte” utilizzo.

	<i>r</i>	<i>r</i>	<i>b</i>	<i>swpd</i>	<i>free</i>	<i>inact</i>	<i>active</i>	<i>si</i>	<i>so</i>	<i>bi</i>	<i>bo</i>	<i>in</i>	<i>cs</i>	<i>us</i>	<i>sy</i>	<i>id</i>	<i>wa</i>	<i>st</i>
<i>r</i>	1,0000	-0,0064	0,0000	0,0543	-0,0412	-0,0349	0,0000	-0,0064	0,0896	0,0848	0,0204	-0,0189	0,0670	0,0469	-0,0577	0,1187	0,0000	
<i>b</i>	-0,0064	1,0000	0,0000	0,1472	-0,0803	-0,1225	0,0000	-0,0033	-0,0062	0,0133	-0,0517	-0,0925	-0,0475	-0,0494	0,0557	-0,0268	0,0000	
<i>swpd</i>	0,0000	0,0000	1,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	
<i>free</i>	0,0543	0,1472	0,0000	1,0000	-0,8048	-0,9759	0,0000	0,2547	0,1346	-0,0485	-0,5688	-0,7030	-0,2509	-0,6437	0,6489	-0,394	0,0000	
<i>inact</i>	-0,0412	-0,0803	0,0000	-0,8048	1,0000	0,8791	0,0000	-0,4035	-0,0570	0,0347	0,2674	0,3487	0,1624	0,4144	-0,4149	0,0317	0,0000	
<i>active</i>	-0,0349	-0,1225	0,0000	-0,9759	0,8791	1,0000	0,0000	-0,3213	-0,1116	0,0483	0,5076	0,6240	0,2438	0,5763	-0,5834	0,0443	0,0000	
<i>si</i>	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	1,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	
<i>so</i>	-0,0064	-0,0033	0,0000	0,2547	-0,4035	-0,3213	0,0000	1,0000	-0,0026	-0,0216	-0,2426	-0,2725	-0,1217	-0,2299	0,2300	-0,2686	0,0000	
<i>bi</i>	0,0000	0,0000	0,0000	0,1346	-0,7030	-0,0000	-0,0026	0,0000	1,0000	-0,0026	-0,0216	-0,2426	0,1425	0,2300	0,1592	0,3764	0,0000	
<i>bo</i>	0,0000	0,0133	0,0000	-0,5688	0,2674	0,5076	0,0000	-0,2426	0,1663	0,0118	0,1213	0,0559	0,1556	0,185	-0,5206	0,0559	0,0000	
<i>in</i>	0,0000	-0,0517	0,0000	-0,5688	0,2674	0,5076	0,0000	-0,2426	-0,1663	0,0118	0,0776	0,1556	0,185	-0,5206	0,0559	0,0000		
<i>cs</i>	-0,0189	-0,0925	0,0000	-0,7030	0,3487	0,6240	0,0000	-0,2725	-0,1048	0,1213	0,8776	1,0000	0,2889	0,7413	-0,7564	0,3324	0,0000	
<i>us</i>	0,0670	-0,0475	0,0000	-0,2500	0,1624	0,2438	0,0000	-0,1217	0,1425	0,0413	0,1556	0,2889	1,0000	0,2833	-0,3872	0,1198	0,0000	
<i>sy</i>	0,0469	-0,0494	0,0000	-0,6437	0,4144	0,5763	0,0000	-0,2299	-0,2130	0,0598	0,5185	0,7413	0,2283	1,0000	-0,9837	0,0132	0,0000	
<i>id</i>	-0,0577	0,0557	0,0000	0,6489	-0,4149	-0,5834	0,0000	0,2386	0,1592	-0,0770	-0,5226	-0,3872	-0,9837	1,0000	-0,0651	0,0000		
<i>wa</i>	0,1187	-0,0268	0,0000	-0,0394	0,0317	0,0443	0,0000	-0,0268	0,3764	0,3126	-0,0509	0,0324	0,1198	0,0132	-0,0651	1,0000	0,0000	
<i>st</i>	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	1,0000	

Figura 3.4: Evoluzione nel tempo della Latency per i tre tipi di pagina

- In seguito alle precedenti considerazioni si è quindi deciso di procedere nell’analisi attraverso l’uso di PCA+Clustering per ottenere un workload sintetico che rappresenti almeno l’80 % della variabilità di quello reale.

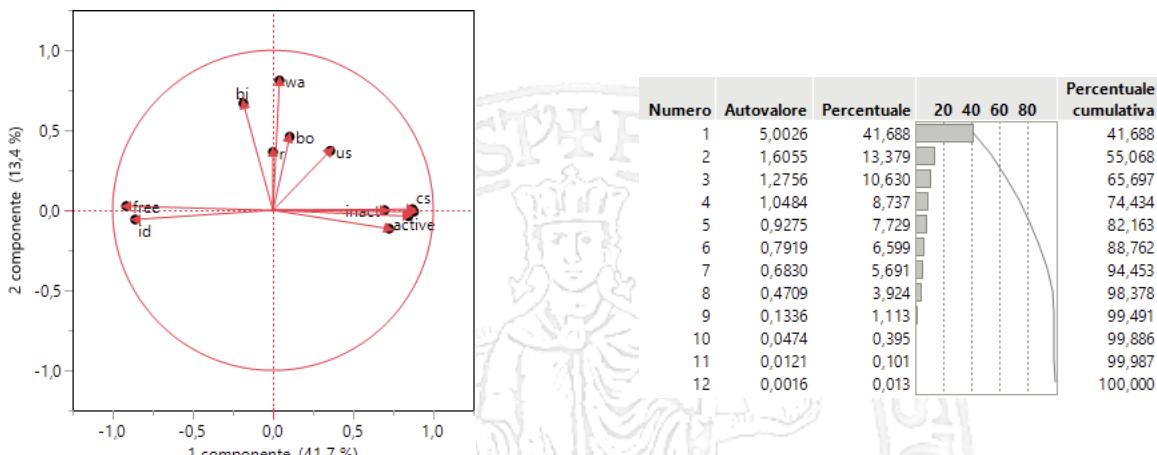


Figura 3.5: Autovalori con percentuale cumulativa

Figura 3.6: Autovalori con percentuale cumulativa

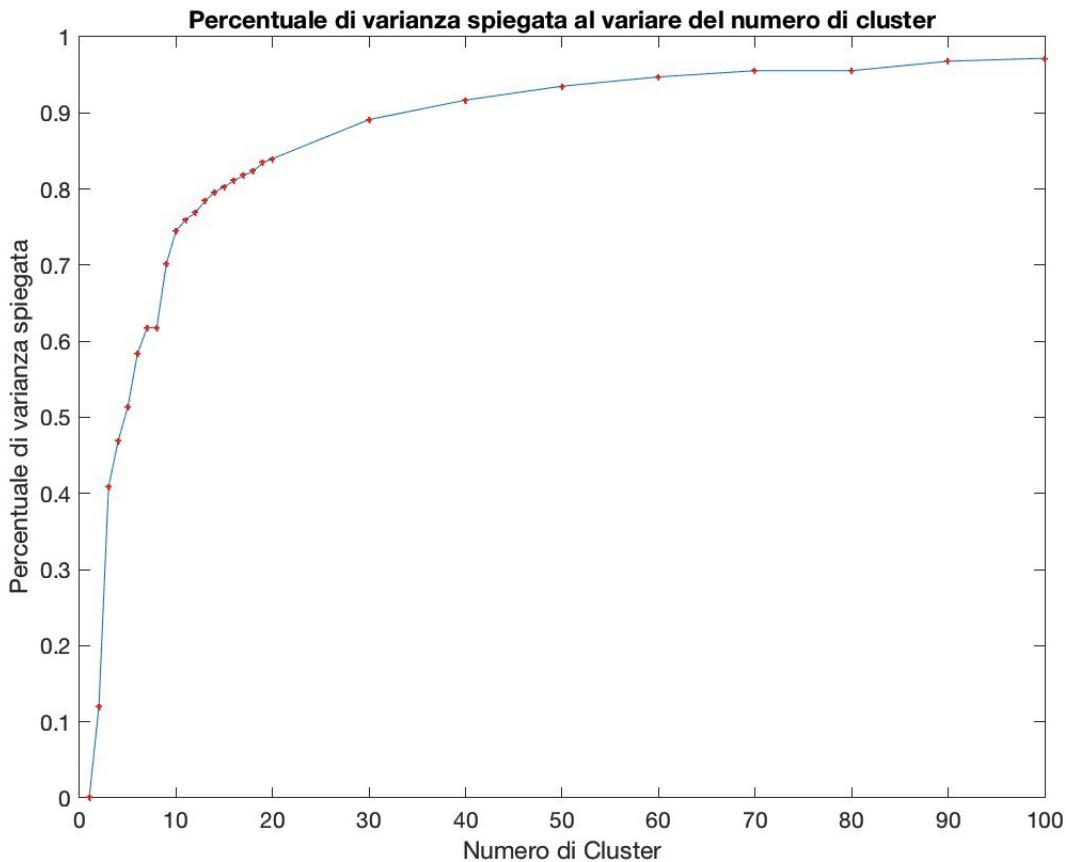


Figura 3.7: Evoluzione nel tempo della Latency per i tre tipi di pagina

Avendo scelto sei componenti principali (conservando una varianza del dataset del 88 %) nella successiva fase di clusterizzazione è stato scelto 30 come numero di cluster per mantenere una percentuale di varianza spiegata superiore al 91 %, per un totale di 80%. Di seguito è riportato in fig. la varianza spiegata al variare del numero di cluster.

### 3.2.3 Web Server Analysis:

Il sistema sotto studio è il server: il servizio offerto dal Server è quello di accettare connessioni e inviare delle pagine html.

L’obiettivo è valutare la produttività di servizio del Server considerando come fattori aggiuntivi i parametri che descrivono lo stato corrente del sistema su cui il server è in esecuzione.

Servizi: Accettare e validare connessioni, e Invio di pagine

Fattori: CPU, System, procs, io, memory and swap parameters, dimensione della pagina da trasferire(piccole, medie, grandi)

Metrica: Throughput

Workload: Richieste HTTP

## 3.3 Capacity Test

Il server è stato sottoposto a quattro tipi di test:

1. Test con pagine piccole (100KB)
2. Test con pagine medie (1MB)
3. Test con pagine grandi (10MB)
4. Test con pagine random tra quelle precedenti

### 3.3.1 Modo di operare

Per scegliere in modo adeguato i carichi a cui sottoporre il sistema, si è scelto il seguente modus operandi.

Oltre al fiuto dell'analista, sono stati utilizzati risultati analitici per orientarsi nello scegliere il giusto numero di utenti nelle misurazioni.

Per avere una stima approssimata del tempo medio di servizio è stato condotto un test in cui vi è un solo utente che invia una richiesta al server immediatamente dopo aver ricevuto una risposta. Dal tempo di servizio medio sarà possibile avere subito indicazione sul throughput massimo.

Il “numero di utenti ottimo”, derivato dal modello analitico, sarà utilizzato per scegliere la risoluzione di campionamento e per individuare approssimativamente il numero di utenti ottimo (nel quale il throughput medio è prossimo a quello di alto carico e il tempo medio di risposta è prossimo a quello di basso carico).

La media è stata scelta come valore rappresentativo di una misurazione poiché risultano di interesse tutte le risposte alle singole richieste.

La media è stata scelta anche come valore rappresentativo delle tre misurazioni, poiché la distribuzione risulta pressoché simmetrica e la media e la mediana coincidono quasi.

### 3.3.2 Capacity test con richieste di pagine di dimensione 100 KB

#### 3.3.2.1 Risultati derivati dal modello analitico

Il tempo stimato per il tempo di servizio è di 1.59 ms. Il throughput massimo risulta essere quindi di 629 hits al secondo.

Il numero di utenti ottimo derivato dal modello analitico è all'incirca di 160 utenti.

$(N^* = 1 + \frac{\text{Tempo medio di richiesta}}{\text{Tempo medio di servizio}} = 1 + \frac{250 \text{ ms}}{2 \text{ ms}} = 160 \text{ utenti})$

Conseguentemente si è scelto inizialmente un passo di campionamento della grandezza delle decine ponendo attenzione al carico di utenti vicino all'ottimo stimato. Successivamente si è campionato nell'ordine delle centinaia per valutare l'evoluzione dei risultati.

Tempo di servizio stimato	1.59 ms
TP massimo stimato	630 ms
Ottimo stimato	160 utenti

Tabella 3.1: Risultati derivati dal modello matematico

### 3.3.2.2 Risultati derivati dalla misurazione

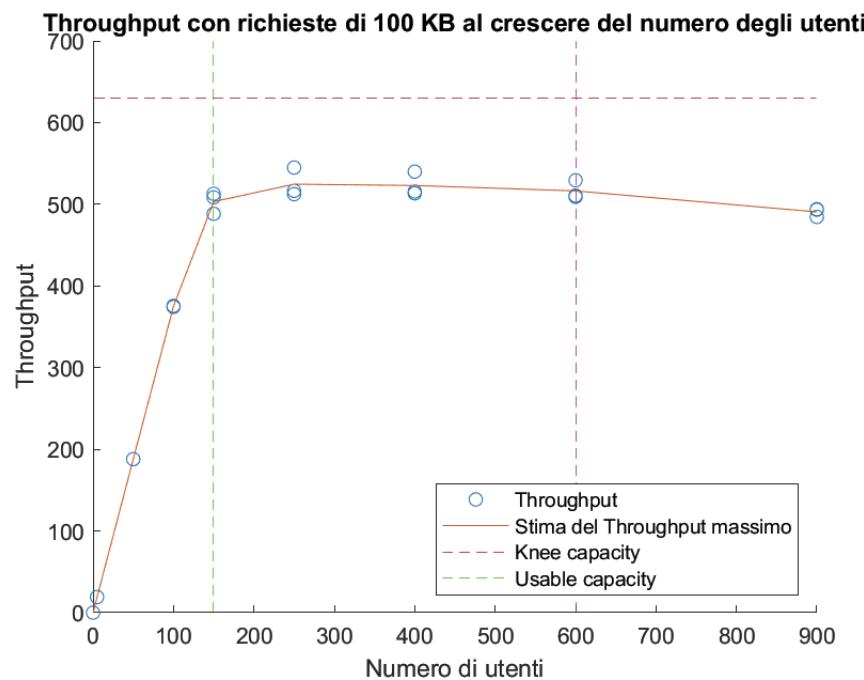


Figura 3.8: Evoluzione del Thorughput con richieste di 100 KB al crescere del numero degli utenti

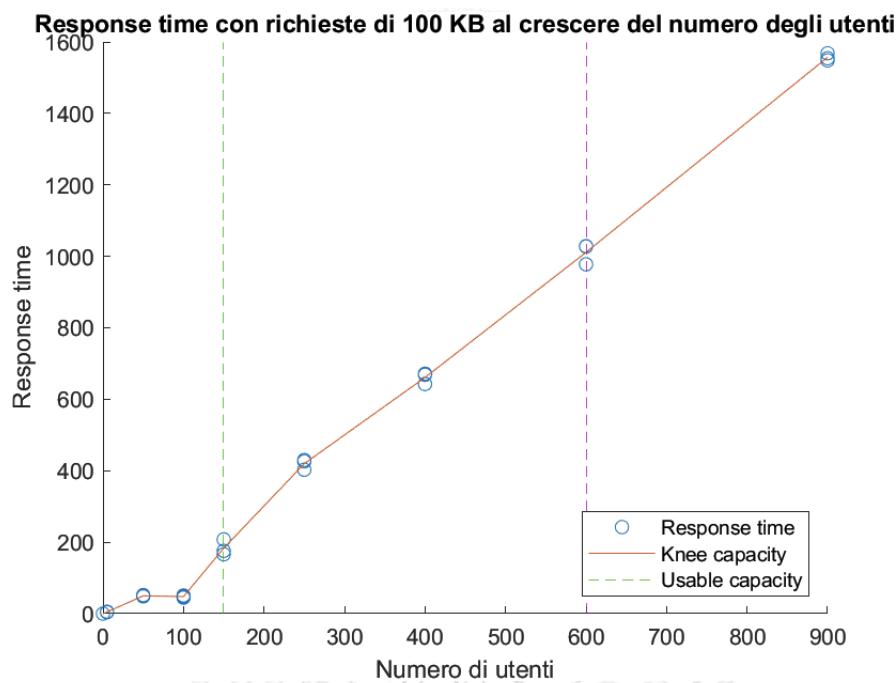


Figura 3.9: Evoluzione del response time con richieste di 100 KB al crescere del numero degli utenti

### 3.3.2.3 Interpretazione dei dati

Come carico ottimale è stato individuato un carico che sia di 150 utenti, in questo punto si raggiunge quasi il massimo throughput raggiungibile (circa 510 hits/s), e il tempo di risposta rimane più che ottimale (circa 170 ms).

Considerando come tempo di risposta massimo accettabile 1 secondo, l'Usable capacity può essere identificata in un carico di 600 utenti.

### 3.3.3 Capacity test con richieste di pagine di dimensione 1 MB

#### 3.3.3.1 Risultati derivati dal modello analitico

Il tempo stimato per il tempo di servizio è di 4.95 ms. Il throughput massimo risulta essere quindi di 202 hits al secondo.

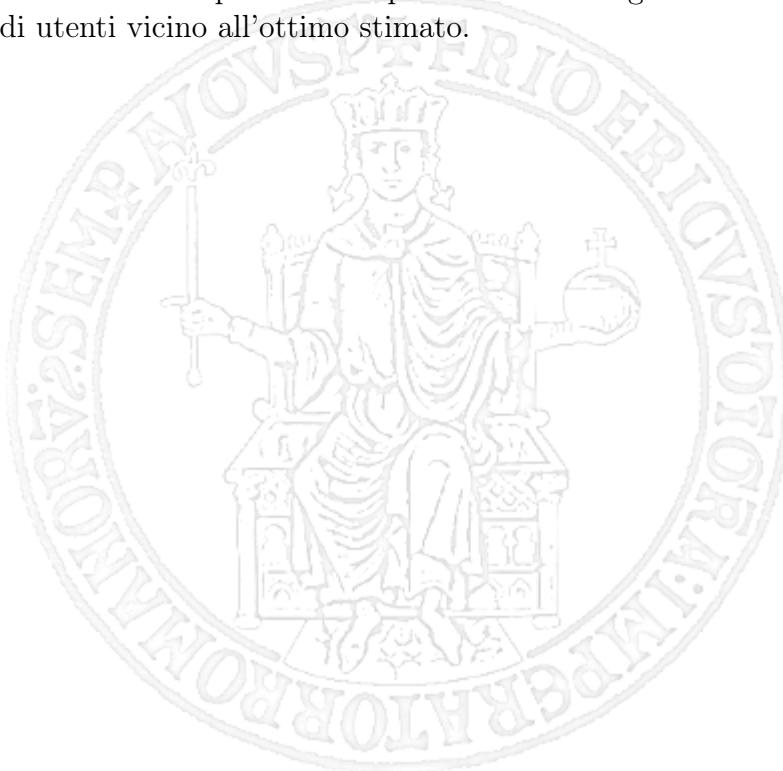
Il numero di utenti ottimo derivato dal modello analitico è all'incirca di 50 utenti.

( $N^* = 1 + \frac{\text{Tempo medio di richiesta}}{\text{Tempo medio di servizio}} = 1 + \frac{250 \text{ ms}}{4.95 \text{ ms}} = 51$  utenti.)

Tempo di servizio stimato	4.95 ms
TP massimo stimato	202 hits al secondo
Ottimo stimato	all'incirca 50 utenti

Tabella 3.2: Risultati derivati dal modello matematico

Conseguentemente si è scelto un passo di campionamento della grandezza delle decine ponendo attenzione al carico di utenti vicino all'ottimo stimato.



### 3.3.3.2 Risultati derivati dalla misurazione

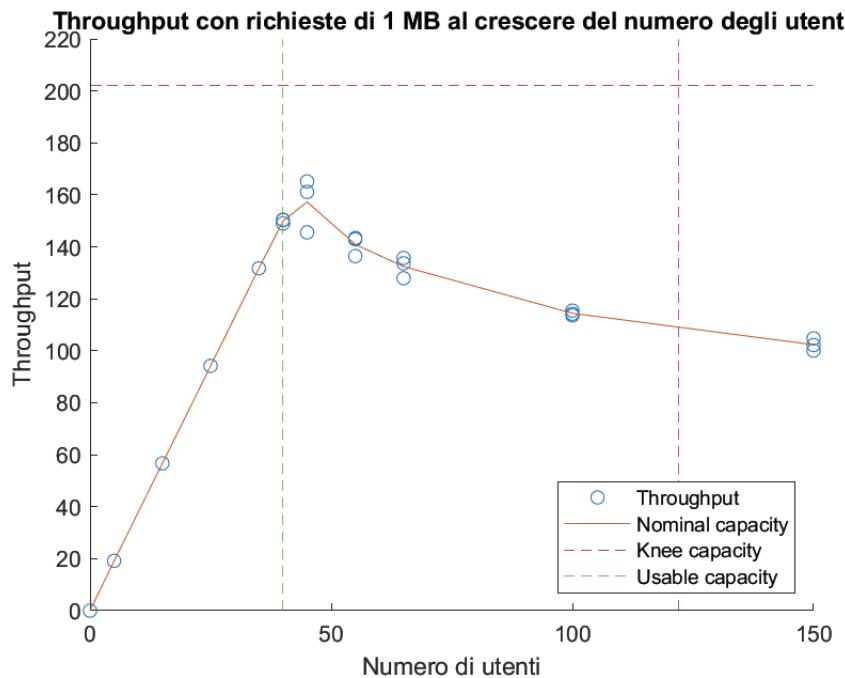


Figura 3.10: Evoluzione del Thorughput con richieste di 1 MB al crescere del numero degli utenti

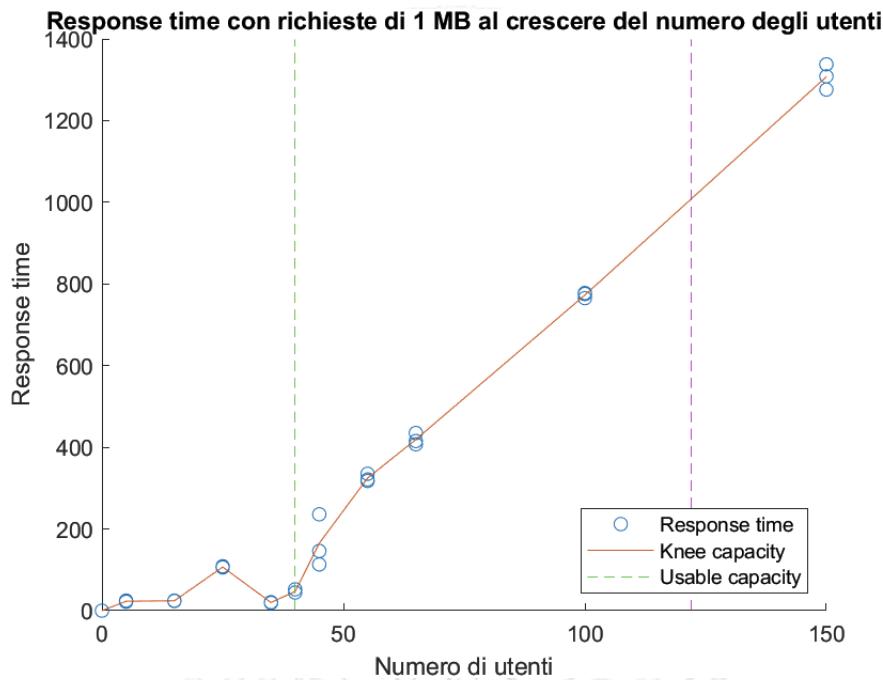


Figura 3.11: Evoluzione del response time con richieste di 1 MB al crescere del numero degli utenti

### 3.3.3.3 Interpretazione dei dati

Come Knee Capacity è stato individuato un carico di 40 utenti, in questo punto il throughput medio è prossimo a quello ottimale (circa 150 hits/s), e il tempo di risposta rimane più che ottimale (circa 47 ms).

Considerando come tempo di risposta massimo accettabile 1 secondo, la Usable capacity può essere identificata in un carico di 122 utenti.

### 3.3.4 Capacity test con richieste di pagine di dimensione 10 MB

#### 3.3.4.1 Risultati derivati dal modello analitico

Il tempo stimato per il tempo di servizio medio è di 32.33 ms. Il throughput massimo risulta essere quindi di 31 hits al secondo.

Il numero di utenti ottimo derivato dal modello analitico è di 9 utenti.

$$N^* = 1 + \frac{\text{Tempo medio di richiesta}}{\text{Tempo medio di servizio}} = 1 + \frac{250 \text{ ms}}{32 \text{ ms}} = 9 \text{ utenti}$$

Tempo di servizio stimato	32.33 ms
TP massimo stimato	31 hits al secondo
Ottimo stimato	9 utenti

Tabella 3.3: Risultati derivati dal modello matematico

#### 3.3.4.2 Risultati derivati dalla misurazione

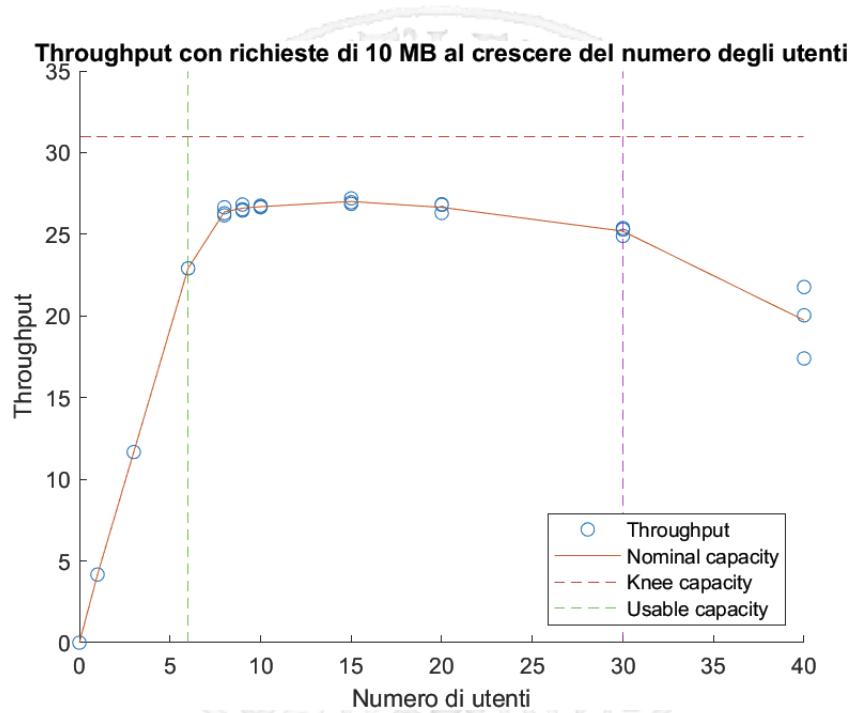


Figura 3.12: Evoluzione del Thorughput con richieste di 10 MB al crescere del numero degli utenti

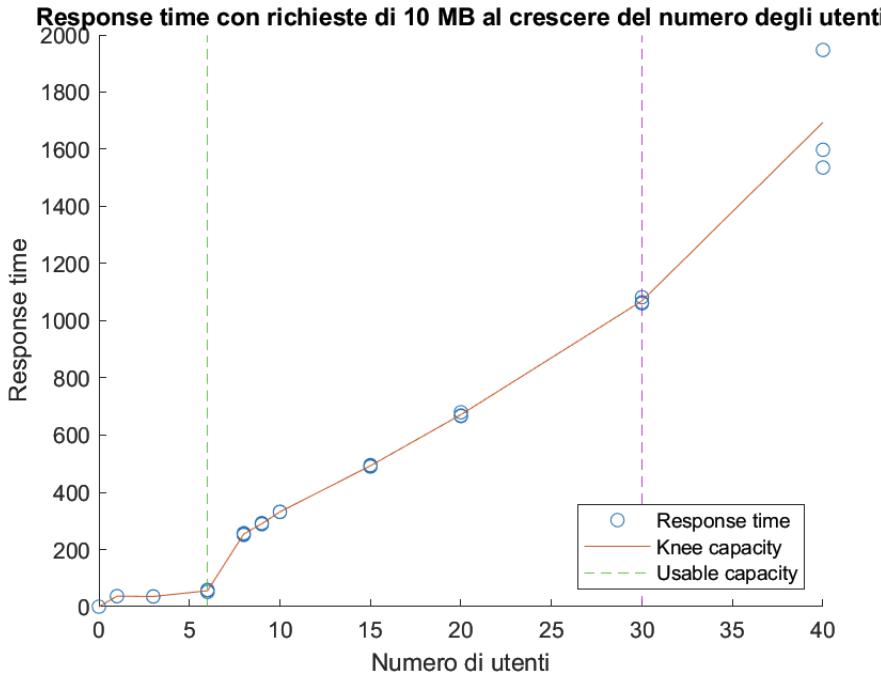


Figura 3.13: Evoluzione del response time con richieste di 10 MB al crescere del numero degli utenti

### 3.3.4.3 Interpretazione dei dati

Come Knee Capacity è stato individuato un carico di 6 utenti, in questo punto il throughput medio è prossimo a quello ottimale (circa 23 hits/s), e il tempo di risposta rimane più che buono.

Impostato come tempo di risposta massimo accettabile 1 secondo, la Usable capacity può essere identificata in un carico di 30 utenti.

## 3.3.5 Capacity test con richieste di pagine di dimensione random

### 3.3.5.1 Risultati derivati dal modello analitico

Il tempo stimato per il tempo di servizio è di 12,75 ms. Il throughput massimo risulta essere quindi di 82 hits al secondo.

Il numero di utenti ottimo derivato dal modello analitico è di 20 utenti.

$(N^* = 1 + \frac{\text{Tempo medio di richiesta}}{\text{Tempo medio di servizio}} = 1 + \frac{250 \text{ ms}}{12.75 \text{ ms}} = 20 \text{ utenti})$

Tempo di servizio stimato	12.75
TP massimo stimato	82 hits al secondo
Ottimo stimato	20 utenti

Tabella 3.4: Risultati derivati dal modello matematico

### 3.3.5.2 Risultati derivati dalla misurazione

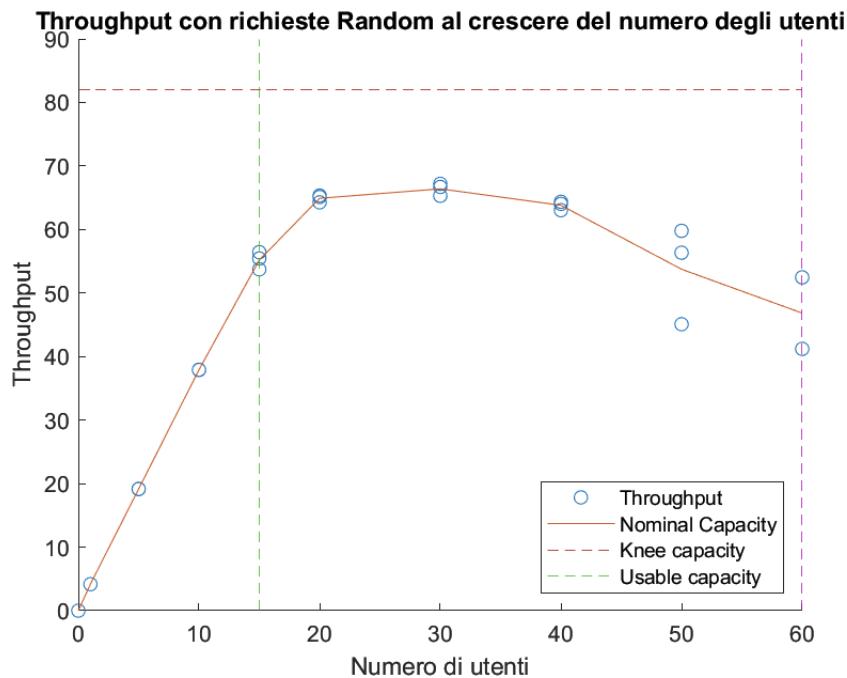


Figura 3.14: Evoluzione del Thorughput con richieste Random al crescere del numero degli utenti

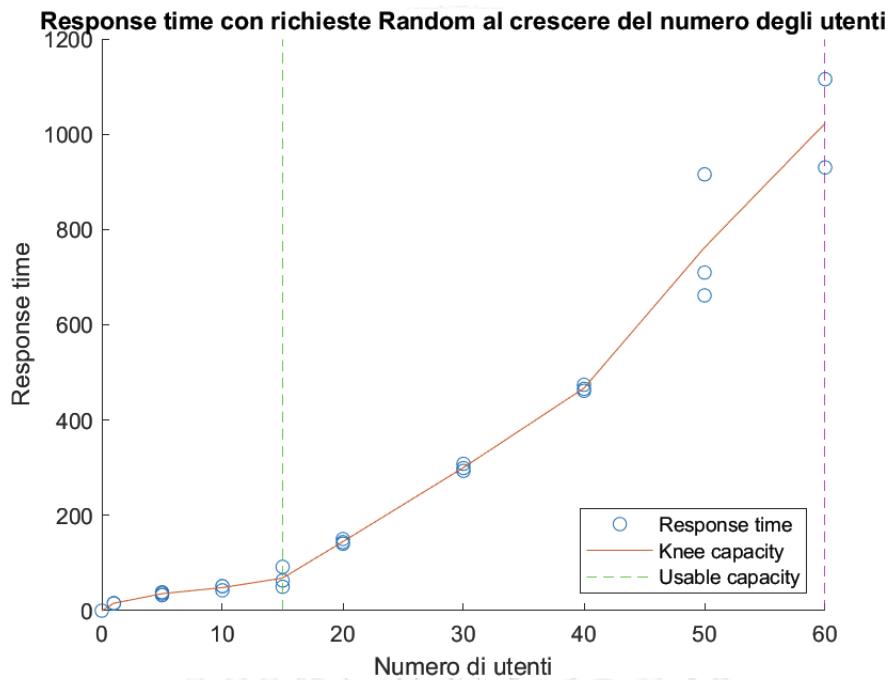


Figura 3.15: Evoluzione del response time con richieste Random al crescere del numero degli utenti

### 3.3.5.3 Interpretazione dei dati

Come Knee Capacity è stato individuato un carico di 15 utenti, in questo punto il throughput medio è prossimo a quello ottimale (circa 55 hits/s), e il tempo di risposta rimane più che buono (70 ms).

Impostato come tempo di risposta massimo accettabile 1 secondo, la Usable capacity può essere identificata in un carico di 60 utenti.

### 3.3.6 Conclusione

	Knee Capacity	Usable Capacity
Test 1 (100 KB)	150	600
Test 2 (1 MB)	40	122
Test 3 (10 MB)	6	30
Test 4 (random)	15	60

Tabella 3.5: Tabella con i range di carico risultanti dai test

I risultati del primo test caratterizzano il caso ottimale in cui il server riesce a raggiungere il massimo throughput ricevendo solo richieste di pagine di piccole dimensioni. I risultati del terzo test rappresentano il caso peggiore in cui il numero di utenti è limitativo. Se si volesse avere la certezza di avere un tempo di risposta mediamente minore di un secondo, dovrebbe essere considerato questo range di carico. I risultati del test 4 caratterizzano il caso medio; questo è il range di carico da prendere in considerazione se si presuppone che non vi sia una priorità di scelta di pagina da parte degli utenti. Chiaramente il range potrebbe essere ampliato se si prendesse in considerazione una maggiore frequenza di occorrenza di pagine piccole.

## 3.4 Design of Experiments

Con le informazioni acquisite in precedenza, progettare con la tecnica del DOE un esperimento per studiare l'impatto delle richieste (fattori) sul response time.

Si è ricorso alla tecnica di DOE Two Factor Experimental Analysis per la valutazione degli effetti che il tipo di pagina e l'intensità di richieste hanno sul tempo di risposta del Server.

I fattori sono stati assunti su due soli livelli in quanto è ragionevole ritenere che al crescere vi sia un degrado delle prestazioni.

In tabella si mostrano innanzitutto i livelli considerati per tali fattori.

Factor Level	Intensity	Page type (B)
Low	10% Usable capacity minima = 3	Piccole (100 KB)
High	90% Usable capacity minima = 27	Medie (1 MB)

Tabella 3.6: Tabella che mostra i livelli considerati per i due fattori Intensity e Page type

### 3.4.1 Modello di regressione

Si suppone che l'effetto dei fattori (singolare e combinato) influisca additivamente sulla risposta, ossia che la risposta sia determinata secondo il seguente modello:

$$y_{ij} = u + q_A * x_{A_i} + q_B * x_{B_i} + q_a q_b x_{A_i} x_{B_i} + e_{ij}$$

Dove  $y_{ij}$  è la risposta osservata per la  $j$ -esima prova dell'esperimento ottenuto ponendo i fattori A e B ai livelli definiti dal trattamento  $i$ -esimo;

$u$  = è la media delle risposte medie dei vari trattamenti;

$e_{ij}$  = l'errore casuale dovuto allo/agli strumenti di misura adoperati, fattori importanti tralasciati in quest'analisi ed altri aspetti ignoti e/o non controllabili.

#### 3.4.1.1 Piano degli esperimenti

Sono state fatte 10 replicazioni delle 4 combinazioni dei livelli risultanti dai due fattori.

### 3.4.2 Analisi dei risultati

#### 3.4.2.1 Modello regressivo stimato:

$$y_{ij} = 6.97825 + 3.06125 * x_{pageType} + 1.90675 * x_{intensity} + 0.44275 x_{pageType} x_{intensity}$$

Il modello esprime il 91 % della variabilità totale (R-squared).

Anche R-squared corretto, una stima non soggetta al problema dell'overfitting, è pari al 90%.

Test degli effetti					
Origine	Nparm	DF	Somma dei quadrati	Rapporto F	Prob > F
Page Type		1	363,91056	245,4907	<.0001*
Intensity		1	145,42782	98,1043	<.0001*
Page Type*Intensity		1	7,84110	5,2895	0,0274*

Analisi della varianza				
Origine	DF	Somma dei quadrati	Media quadratica	Rapporto F
Modello	3	517,17949	172,393	116,2948
Errore	36	53,36569	1,482	Prob > F
C. totale	39	570,54518		<.0001*

Figura 3.16: Devianze dei vari fattori, devianza totale e devianze dei residui

#### 3.4.2.2 Importanza dei parametri

- Page Type = 65%
- Intensity = 25%

- Interazione= 1 %
- Errore = 9 %

Il tipo di pagina spiega la maggior parte della variabilità della response time osservata, a seguire vi è l'intensità del carico.

L'effetto dell'interazione ha un'importanza relativamente bassa e ciò è abbastanza evidente dal grafico dei profili di interazione.

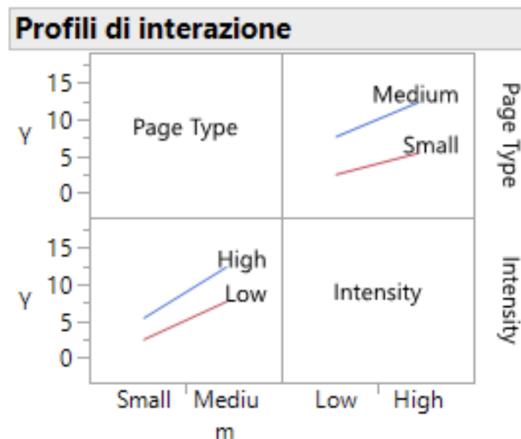


Figura 3.17: Profili di interazione

Le rette sono quasi parallele: la differenza tra il tempo di risposta di un pacchetto small con un carico low con quello con carico high è leggermente più piccola dell'analogia differenza avendo considerato un pacchetto high. La differenza però c'è e potrebbe essere considerata nello spiegare meglio la variabilità della response time.

Prima di fare affidamento agli intervalli di confidenza e alla significatività statistica indicata da JMP, è necessario verificare l'ipotesi di normalità e di omoschedasticità degli errori.

### 3.4.2.3 Ipotesi di normalità: rigettata

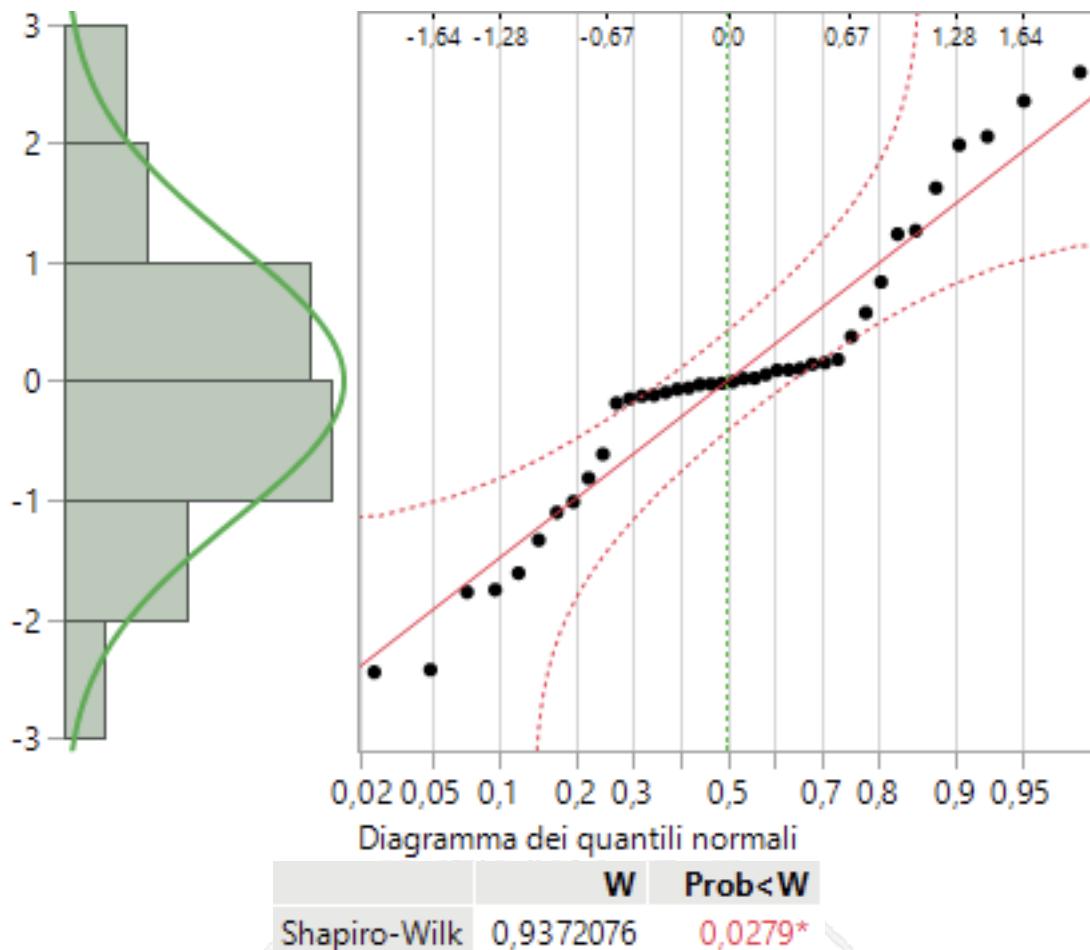


Figura 3.18: Plot quantile quantile e test di normalità Shapiro Wilk

L'ipotesi nulla è che i residui abbiano distribuzione normale.

Il p-value è minore del livello di significatività scelto precedentemente del 0.05, l'evidenza empirica è fortemente contraria all'ipotesi nulla che quindi va rifiutata.

### 3.4.2.4 Ipotesi di omoschedasticità: rigettata

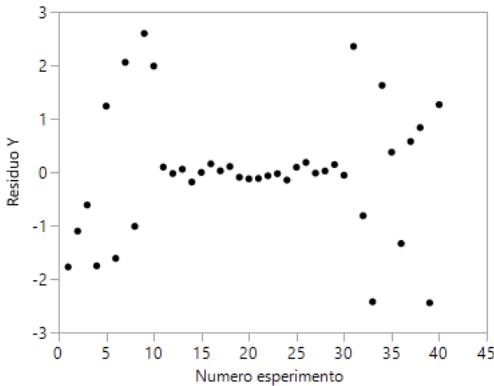


Figura 3.19: Plot residui - numero esperimento

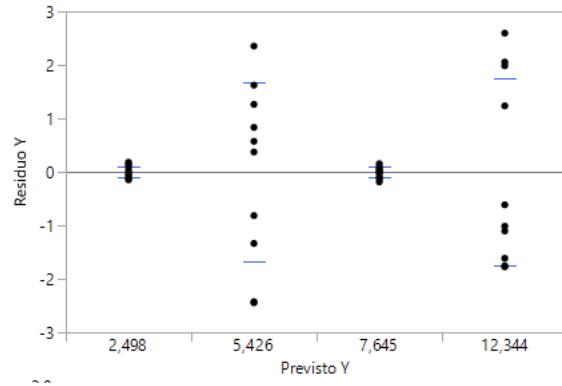


Figura 3.20: Plot residui - stime di y

Test	Rapporto F	Num DF	Den DF	Prob > F
O'Brien[.5]	8,9831	3	36	0,0001*
Brown-Forsythe	8,8060	3	36	0,0002*
Levene	27,3976	3	36	<,0001*
Bartlett	23,7179	3	.	<,0001*

Figura 3.21: Test con ipotesi nulla l'omoschedasticità

L'ipotesi nulla è che la varianza dei residui è costante.

Essendo il p-value minore dello 0.05, il test di ipotesi nulla è rigettato.

### 3.4.2.5 Significatività dei fattori

Per valutare la significatività dei fattori è stato applicato il test non parametrico heteroschedastico Kruskal-Wallis.

Prendendo come esempio il fattore di Pagetype, le response time osservate e i residui sono stati relazionati ai due livelli Small e medium del fattore.

L'ipotesi nulla del test è che l'effetto dei due livelli è equivalente, in altre parole che non vi è significativa differenza nei dati tra i due rispettivi gruppi.

Tale ipotesi non è rigettata nello spiegare i residui, ciò significa che la variazione dei dati tra i due livelli non è spiegata dall'errore.

Complementarmente, per quanto riguarda il response time l'ipotesi nulla è rigettata, i due effetti risultano significativamente differenti, ciò significa che il fattore page type è significativo, ovvero il suo coefficiente è significativamente differente da zero.

Analogamente lo stesso ragionamento è stato fatto per il fattore Intensity.

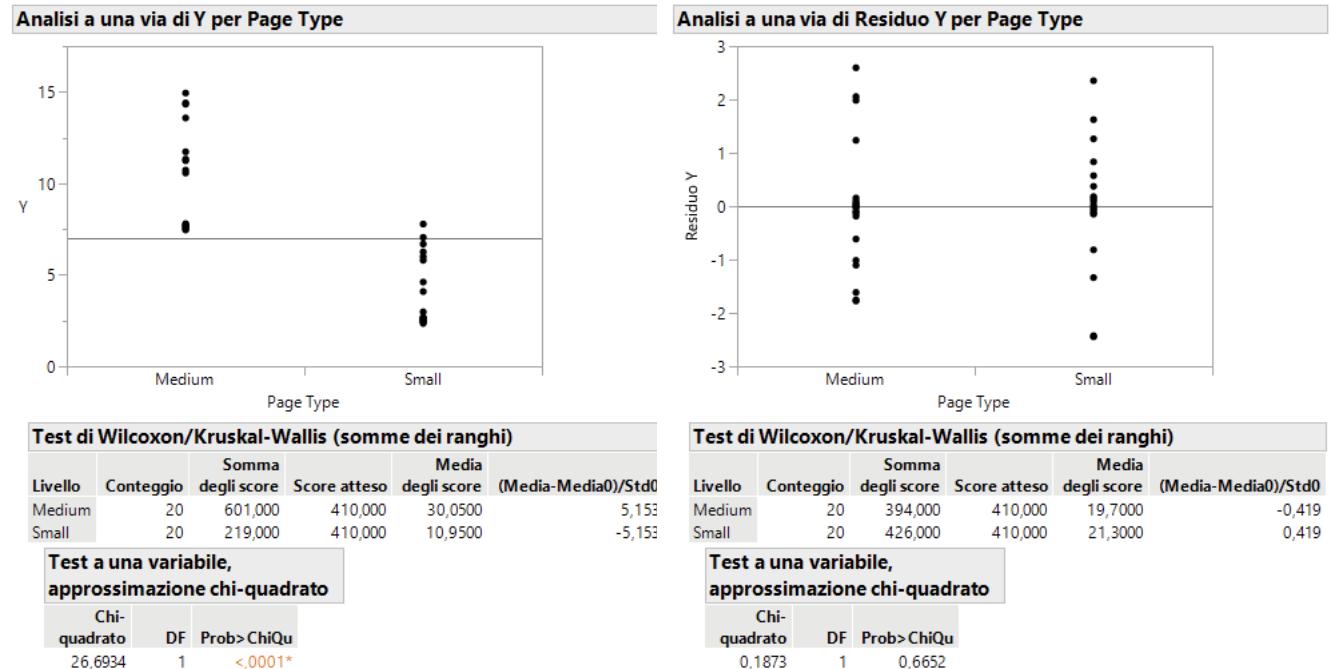


Figura 3.22: Plot Y osservate - Page Type

Figura 3.23: Plot residui - Page Type

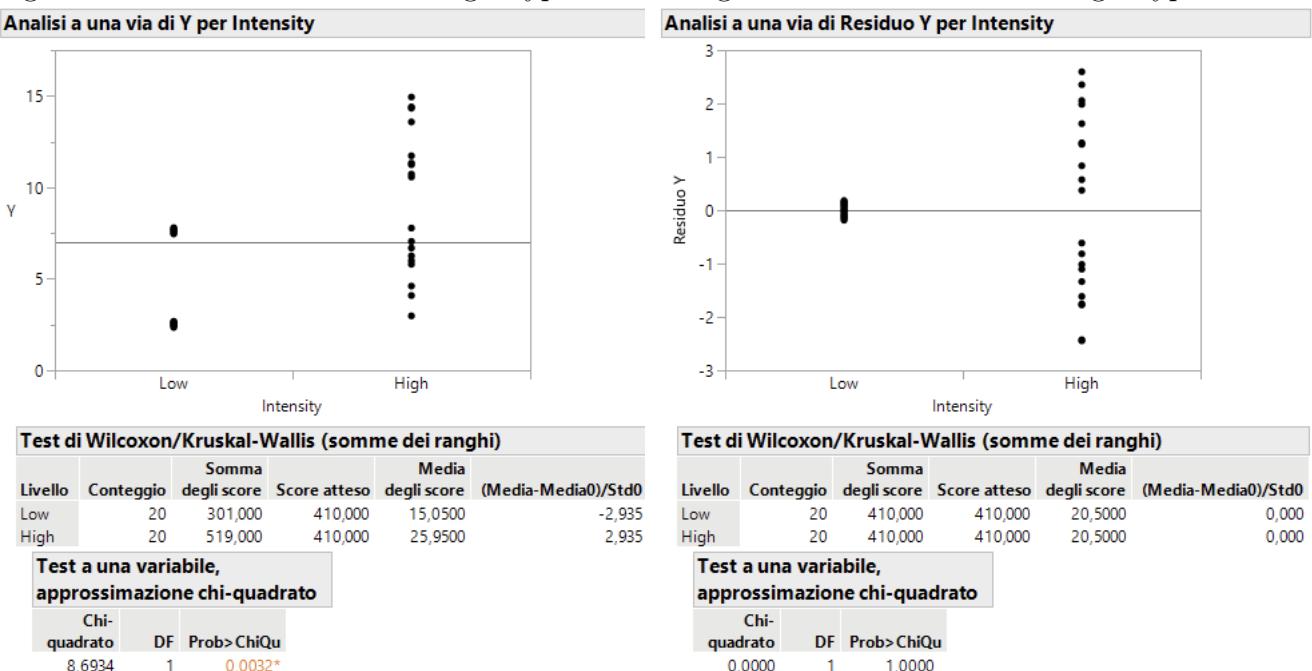


Figura 3.24: Plot Y osservate - Intensity

Figura 3.25: Plot residui - Intensity

# Capitolo 4

## Dependability

### 4.1 Esercizio 01

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

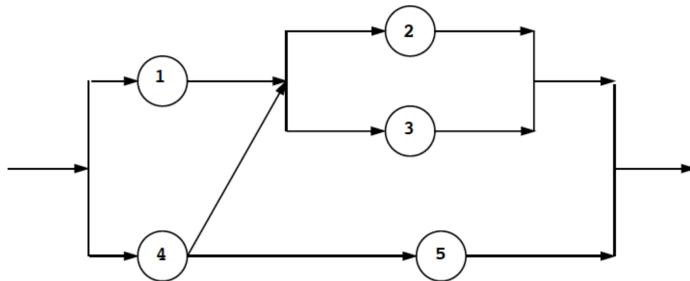


Figura 4.1: RDB del problema

#### 4.1.1 Soluzione: conditioning

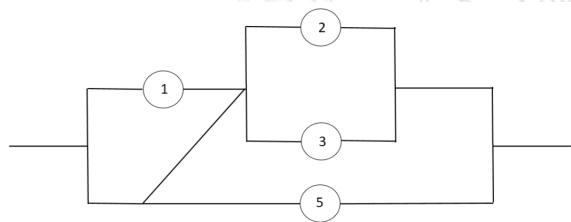


Figura 4.2: RDB con nodo 4 che non è fallito

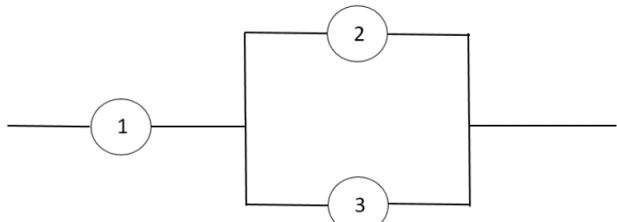


Figura 4.3: RDB con nodo 4 che è fallito

Il problema è un non serie-paralello. Per risolvere tale problema si ricorre al conditioning. Il nodo 4 è stato scelto per partizionare lo spazio campione in due insiemi disgiunti in quanto non ci permette di vedere il sistema in serie o in parallelo.

I casi che quindi analizziamo sono :

$$P(\text{sistema funziona}) = P(\text{sistema funziona} | 4 \text{ fallito})P(4 \text{ fallisce}) + P(\text{sistema funziona} | 4 \text{ non fallito})P(4 \text{ non fallisce})$$

1. Probabilità che il sistema funziona dato che il componente 4 funziona ( si sostituisce il componente con un corto circuito):

$$P(\text{sistema funziona} | 4 \text{ funziona}) = 1 - (1 - R_{2||3})(1 - R_5) = R^3 - 3R^2 + 3R$$

2. la Probabilità che il sistema funziona dato che il componente 4 non funziona (si sostituisce il componente con

un circuito aperto):

$$P(\text{sistema funziona} | 4 \text{ non funziona}) = R(1 - (1 - R)^2) = -R^3 + 2R^2$$

Calcolo della Reliability dell'intero sistema:

$$P(\text{sistema funziona}) = (-R^3 + 2R^2) * (1 - R) + (R^3 - 3R^2 + 3R) * R = 2R^4 - 6R^3 + 5R^2$$

Avendo calcolato la Reliability dell'intero sistema si può passare al calcolo della MTTF, sapendo

che il fallimento di ogni componente è considerato come un esponenziale:

$$MTTF = \int_0^\infty (2R^4 - 6R^3 + 5R^2)dt = \int_0^\infty (2e^{-4\lambda t} - 6e^{-3\lambda t} + 5e^{-2\lambda t})dt = \frac{1}{2\lambda} - \frac{2}{\lambda} + \frac{5}{2\lambda} = \frac{1}{\lambda}$$

## 4.2 Esercizio 02

Vogliamo confrontare due diversi modi di utilizzare la ridondanza per incrementare la reliability di un sistema. Supponiamo che il sistema abbia bisogno di s componenti identici in serie per funzionare in maniera appropriata. Supponiamo inoltre che ci vengano dati m\*s componenti. Tra i due schemi, quale fornisce una maggiore reliability? Supposta R la reliability del singolo componente, derivare le espressioni delle reliability delle due configurazioni. Comparare le espressioni per m = 3 e s = 3.

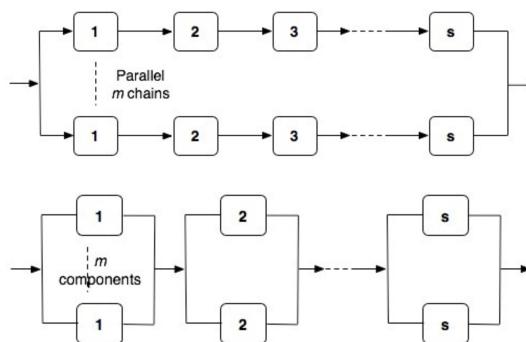


Figura 4.4: RDB dei due sistemi a confronto

### 4.2.1 Soluzione

Nella gura sopra mostrata sono presenti due RBD, quello nella parte superiore non è altro che un parallelo di serie, mentre quello nella parte inferiore è una serie di paralleli. Ciò che si vuole capire è quale dei due grantsce una Reliability più alta. A ciò potremmo rispondere in modo molto rapido considerando i success path. Si noti che i success path del primo RBD sono solamente m mentre

quello del secondo sono  $m^s$ . Per dimostrare quanto detto procediamo al calcolo della Reliability dei due RBD. Definiremo la Reliability del primo RBD  $R_{ps}$  mentre quella del secondo  $R_{sp}$ :

$$R_{ps} = 1 - (1 - R^s)^m$$

$$R_{sp} = (1 - (1 - R)^m)^s$$

sostituendo nelle espressioni  $m=s=3$

$$R_{ps} = 1 - (1 - R^3)^3 = R^9 - 3R^6 + 3R^3$$

$$R_{sp} = (1 - (1 - R)^3)^3 = R^9 - 9R^8 + 36R^7 - 81R^6 + 108R^5 - 81R^4 + 27R^3$$

## Esercizio 03

In gara è mostrata l'architettura di una rete di computer in un sistema bancario. L'architettura è chiamata skip-ring network ed è progettata per permettere ai processori di comunicare anche dopo l'avvenimento di un failure in un nodo. Ad esempio, se il nodo 1 fallisce, il nodo 8 può bypassare il nodo fallito in stradando i dati sul link alternativo che collega il nodo 8 con il 2. Assumendo che i link siano perfetti e i nodi abbiano ognuno una reliability  $R_m$ , derivare l'espressione per la reliability della rete. Se  $R$  segue la legge di fallimento esponenziale e il failure rate di ogni nodo è di 0.005 failure all'ora, determinare la reliability del sistema alla ne di un periodo di 48 ore.

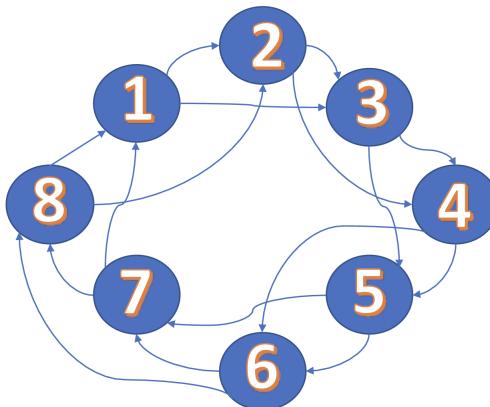


Figura 4.5: Skip-ring network

### 4.2.2 Soluzione

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

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

Tuttavia essa non risulterebbe essere corretta, in quanto non tiene in considerazione l'adiacenza dei nodi falliti. Quindi per applicarla deve essere leggermente modicata, ossia bisogna sottrarre ai vari risultati del coefficiente binomiale della sommatoria, le configurazioni di nodi consecutivi, che indicheremo con  $c_i$ , che porteranno al fallimento (nonostante non siano falliti più di 4 nodi). La formula modicata del M-out-of-N diventa quindi:

$$\sum_{i=0}^{N-M} \left( \binom{N}{i} - c_i \right) R^{N-i} (1-R)^i$$

Essendo che la sommatoria varia tra 0 e  $N-M = 4$ , dobbiamo escludere le configurazioni in cui falliscono 2, 3 e 4 nodi consecutivi:

- configurazioni in cui falliscono 2 nodi consecutivi: 8;
- configurazioni in cui falliscono 3 nodi consecutivi : 8 a cui vanno sommate le configurazioni in cui falliscono due nodi consecutivi più uno qualsiasi (che non sia adiacente alla coppia, perchè ricade nel caso in cui falliscono 3 nodi consecutivi). Le coppie sono 8 e i nodi non adiacenti che possono fallire sono 4, quindi 32 configurazioni. In definitiva le configurazioni da escludere in questa casistica sono  $8 + 32 = 40$ ;
- configurazioni in cui falliscono 4 nodi consecutivi: in questa casistica è più facile considerare i casi in cui il sistema non fallisce, ossia quando sono attivi tutti i nodi pari o quando sono attivi tutti i nodi dispari. Quindi il numero di configurazioni da escludere è:  $\binom{8}{4} - 2 = 68$

Quindi ci = [0 0 8 40 68]

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

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

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

$$R_{SYS}(48) = -e^{-8*0.005*48} + 8e^{-7*0.005*48} - 16e^{-6*0.005*48} + 8e^{-5*0.005*48} + 2e^{-4*0.005*48} = 0.7289$$

Quindi la probabilità che il sistema non fallisca nell'intervallo di tempo da 0 a 48 ore è pari a 0.7289

## 4.3 Esercizio 04

Un'applicazione richiede in un sistema multiprocessore che almeno tre processori debbano essere disponibili con probabilità maggiore del 99%. Il costo di un processore con una reliability dell'80% è di 1000\$, e ogni incremento del 10% di reliability costerà 500\$. Determinare il numero di processori (n) e la reliability (p) di ogni processore (assumendo che i processori abbiano la stessa reliability) che minimizzino il costo totale del sistema.

### 4.3.1 Soluzione

Per risolvere tale problema è stata calcolata la reliability 3-out-of-n al variare del numero di processori n nel caso in cui siano scelti processori con reliability pari 0.80, 0.88, 0.968.

E' stato usato un semplice script che plotta la CMF(N-3) della distribuzione binomiale Bin(N=Numero processori, UnReliability=p).

In fig è mostrata l'evoluzione delle reliability del sistema per i tre tipi di reliability dei singoli processori al variare del numero dei processori .

Da tale plot è stato individuato il numero di processori che permettono di raggiungere la soglia 0.99 di reliability del sistema.

Se la reliability di ogni singolo processore è 0.80 sono necessari almeno 7 processori per raggiungere la soglia 0.99 e quindi con un costo complessivo di 7000 \$ .

Se la reliability di ogni singolo processore è 0.88 sono necessari almeno 6 processori per raggiungere la soglia 0.99 e quindi con un costo complessivo di 9000 \$.

Se la reliability di ogni singolo processore è 0.968 sono necessari almeno 4 processori per raggiungere la soglia 0.99 e quindi con un costo complessivo di 8000\$.

```

1 % programmazione lineare Esercizio
2 % Dependability
3
4 %vettore
5 valori_Proc_rel_min = zeros (1,7);
6
7 for i = 3:10
8     pd = makedist('Binomial','
9         N',i,'p',0.20);
10    valori_Proc_rel_min(i-2) =
11        pd.cdf(i-3);
12
13 end
14
15 valori_Proc_rel_medio = zeros
16 (1,7);
17
18 for i = 3:10
19     pd = makedist('Binomial','
20         N',i,'p',0.12);
21     valori_Proc_rel_medio(i-2) =
22         pd.cdf(i-3);
23
24 end
25
26 valori_Proc_rel_max = zeros (1,7);
27
28 for i = 3:10
29     pd = makedist('Binomial','
30         N',i,'p',1-0.968);
31     valori_Proc_rel_max(i-2) =
32         pd.cdf(i-3);
33
34 end

```

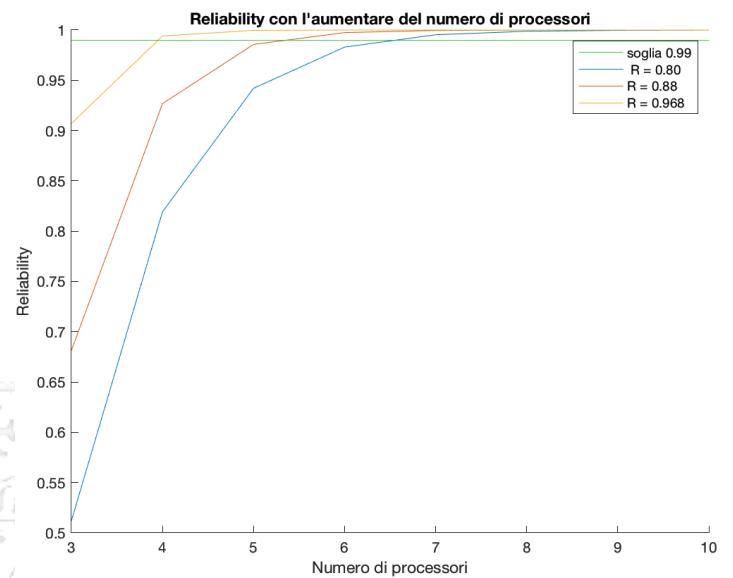


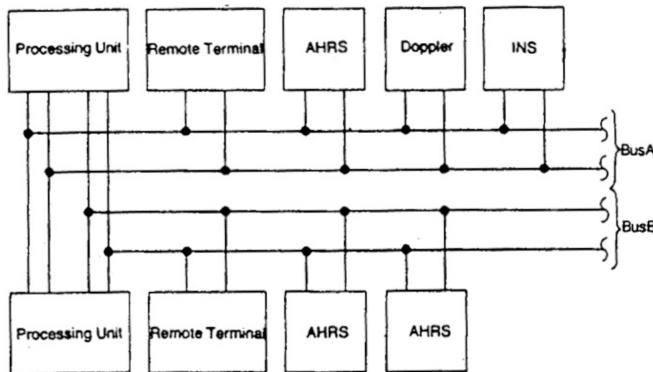
Figura 4.7: RDB con nodo 4 che è fallito

Figura 4.6: RDB con nodo 4 che non è fallito

## 4.4 Esercizio 05

Il sistema mostrato in figura è un sistema di elaborazione per un elicottero. Il sistema ha una ridondanza duale sia per i processori che per le unità di interfacciamento. Vengono utilizzati due bus, ed ogni bus è replicato.

La parte interessante del sistema è l'equipaggiamento di navigazione. Il velivolo può essere completamente guidato usando l'Inertial Navigation System (INS). Se tale INS fallisce, il velivolo può essere guidato attraverso una combinazione di Doppler e AHRS (Altitude Heading Reference System). Di quest'ultima unità, ne sono presenti 3, delle quali una sola è necessaria. I dati dal Doppler e un AHRS possono essere usati in sostituzione del componente INS se esso fallisce. A causa di altri sensori e strumentazioni, sono richiesti entrambi i bus affinché il sistema funzioni in maniera appropriata, indipendentemente dal sistema di navigazione utilizzato.



Equipment	MTTF (hr)
Processing Unit	5000
Remote Terminal	2500
AHRS	1000
INS	1000
Doppler	300
Bus	10000

Figura 4.9: tabella MTTF dei componenti

Figura 4.8: Architettura velivolo

1. Disegnare l'RBD del sistema;
2. Disegnare il Fault Tree del sistema ed analizzare i minimal cutset;
3. Calcolare la reliability per un'ora di volo utilizzando i valori di MTTF in tabella. Assumere che venga applicata la legge di fallimento esponenziale e che la fault coverage sia perfetta
4. Ripetere il punto precedente, ma stavolta incorporare un fattore di coverage per la fault detection e ricongurazione delle unità di elaborazione. Usando gli stessi dati di fallimento, determinare il valore approssimativo di fault coverage richiesto per ottenere (alla ne dell'ora) una reliability di 0.9999.

## 4.4.1 Soluzione

### 4.4.1.1 Quesito 1

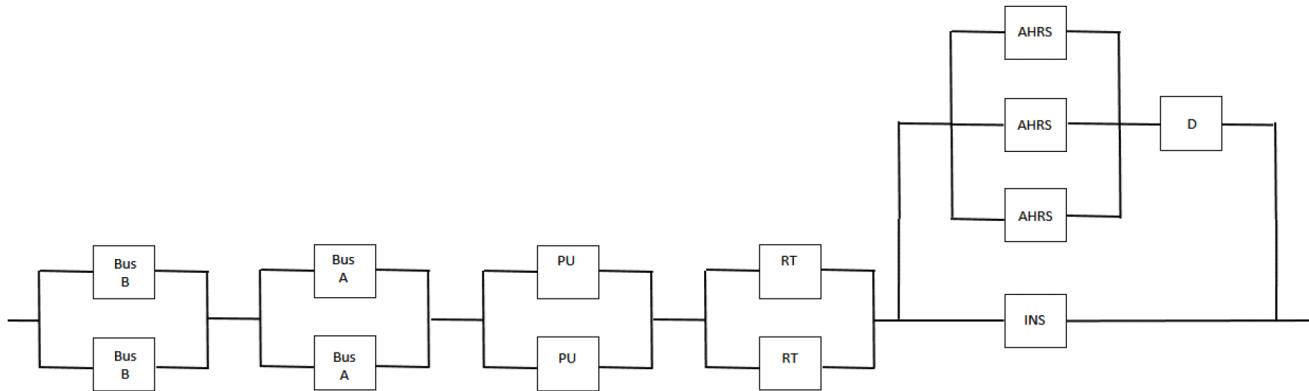


Figura 4.10: RBD

### 4.4.1.2 Quesito 2

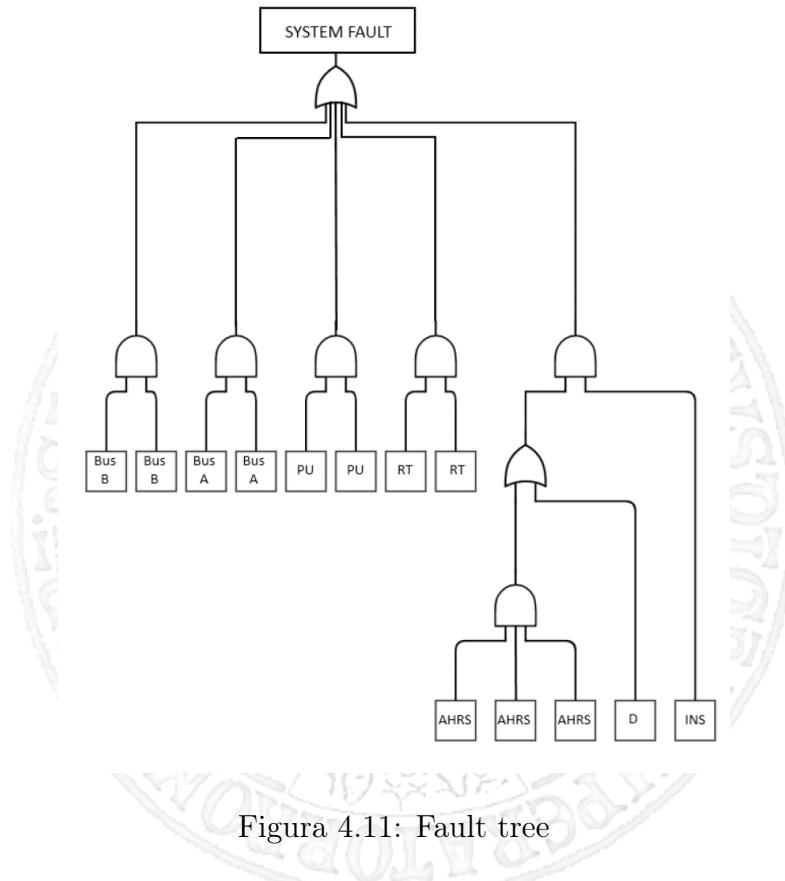


Figura 4.11: Fault tree

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

$$F = (B1*B2) + (A1*A2) + (PU1*PU2) + (RT1*RT2) + AHRS1*AHRS2*AHRS3*INS + D*INS$$

Quindi i minimal cut-set sono:

- Bus B1 e B2, ossia il fallimento di entrambi i bus B;
- Bus A1 e A2, ossia il fallimento di entrambi i bus A;
- Processing Unit PU1 e PU2, ossia il fallimento di entrambe le Processing Unit;
- Remote Terminal RT1 e RT2, ossia il fallimento di entrambi i terminali remoti;
- AHRS1; AHRS2; AHRS3; D; INS, ossia il fallimento dell'Inertial Navigation System contemporaneamente al fallimento del Doppler, oppure il fallimento dell'Inertial Navigation System contemporaneamente al fallimento di tutti gli Altitude Heading Reference System.

#### 4.4.1.3 Quesito 3

Per analizzare la Reliability del velivolo dopo un'ora di volo è necessaria una previa analisi delle Reliability dei singoli componenti, trovando il failure rate  $\lambda$  di ognuno di essi:

- $\lambda_{PU} = \frac{1}{MTTF_{PU}} = \frac{1}{5000} = 0.0002$
- $\lambda_{RT} = \frac{1}{MTTF_{RT}} = \frac{1}{2500} = 0.0004$
- $\lambda_{AHRS} = \frac{1}{MTTF_{AHRS}} = \frac{1}{1000} = 0.001$
- $\lambda_{INS} = \frac{1}{MTTF_{INS}} = \frac{1}{1000} = 0.001$
- $\lambda_D = \frac{1}{MTTF_D} = \frac{1}{300} = 0.003$
- $\lambda_{BUS} = \frac{1}{MTTF_{BUS}} = \frac{1}{10000} = 0.0001$

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

$$R_{SYS} = (1 - (1 - R_{BUS})^2)^2 (1 - (1 - R_{PU})^2) (1 - (1 - R_{RT})^2) \{1 - [(1 - R_{INS})((1 - (1 - R_{AHRS})^3)R_D)]\}$$

Calcolandolo con matlab si è riuscito ad ottener ei seguenti risultati mostrati in figura.

```
Rbus =
0.99999980002000
Rrt =
0.999999840063985
Rahrs =
0.99999999001499
Rd =
0.997004495503373
Rins =
0.999000499833375

Rpu =
0.99999960007999
Rsys =
0.999996786066414
```

Figura 4.12: Reliability dei singoli componenti del sistema dopo un'ora di volo

#### 4.4.1.4 Quesito 4

Supponendo di introdurre un circuito di detection e switching per le CPU, si ricalcola la Reliability del parallelo dei componenti delle CPU. Si introduce il parametro c che denisce la probabilità di rilevare un fallimento e di effettuare correttamente lo switching. La nuova Reliability del parallelo delle CPU diventa quindi:

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

$$R_{SYS} = (1 - (1 - R_{BUS})^2)^2(1 - (1 - R_{RT})^2)\{1 - [(1 - R_{INS})((1 - (1 - R_{AHRS})^3)R_D)]\} * (R_{PU} + c(1 - R_{PU})R_{PU})$$

Dovendo soddisfare il vincolo che la Reliability del sistema in un'ora deve essere maggiore di 0.9999, possiamo ricavarci il parametro c nel modo seguente:

$$R_{SYS} > 0.9999$$

$$c > \frac{0.9999 - R_{PU} * (1 - (1 - R_{BUS})^2)^2(1 - (1 - R_{RT})^2)\{1 - [(1 - R_{INS})((1 - (1 - R_{AHRS})^3)R_D)]\}}{(1 - R_{PU})R_{PU} * (1 - (1 - R_{BUS})^2)^2(1 - (1 - R_{RT})^2)\{1 - [(1 - R_{INS})((1 - (1 - R_{AHRS})^3)R_D)]\}}$$

I risultati ottenuti sono mostrati nella figura sottostante

```
Rbus =
0.999999980002000
Rrt =
0.999999840063985
Rahrs =
0.99999999001499
Rd =
0.997004495503373
Rins =
0.999000499833375

Rpu =
0.999800019998667
c =
0.515922931423784
```

Figura 4.13: Reliability dei singoli componenti del sistema dopo un'ora di volo e valore del parametro c

Si può quindi dedurre che per ottenere una Reliability del 99,99%, il valore minimo del parametro c deve essere circa 0.516.

# Capitolo 5

## FFDA

### 5.1 Introduzione

La FFDA è una tecnica utilizzata per stimare gli attributi di dependability attraverso misurazioni su un sistema deploiauto sotto le condizioni di un workload reale.

La FFDA è un approccio sistematico e come tale ha dei passi definiti:

- Data logng and collection
- filtering
- manipulation
- data analysis

Nel seguente elaborato si hanno a disposizione già gli events log filtrati e ci si concentrerà sulle successive fasi.

### 5.2 FFDA: Mercury

Il sistema Mercury è un cluster composto da nodi IBM organizzati in tre livelli: nodi login (gestiscono il front-end verso gli utenti), nodi computation e nodi storage. Tutti sono gestiti da un unico nodo di management: tg-master. Per quanto riguarda il log del sistema (MercuryErrorLog.txt) è costituito dai seguenti campi:

- timestamp;
- nodo origine;
- categoria dell'errore: DEV, PRO, MEM, I-O, OTH;
- messaggio di errore;

I log events di input da cui si parte con questo elaborato è già filtrato (solo errori relativi a fallimenti letali).

#### 5.2.1 Traccia

1. Condurre l'analisi per MercuryErrorLog.txt:

- plottare il tuple count e selezionare una CWIN;

- raggruppare le entries utilizzando la CWIN;
  - distribuzione della Reliability empirica;
  - t della Reliability empirica (provare tutti i modelli: esponenziale, iper-esponenziale, weibull);
  - analizzare i modelli risultanti con il K-S test.
2. Selezionare i primi 5 nodi con maggiori entries; determinare per ogni nodo CWIN e numero di tuple;
- i nodi hanno tutti la stessa CWIN?
  - esistono dei colli di bottiglia per la Dependability, cioè dei nodi con un gran numero di tuple rispetto agli altri?
  - plottare la Reliability per i nodi con un numero di campioni di interarrivals significativamente grande ( $>30$ )
3. Per ogni categoria (esclusa OTH) determinare CWIN, tuple count e modello di Reliability:
- qual è la categoria con il maggior numero di tuple?
  - quale categoria è più/meno Reliable?
4. Sui nodi funzionalmente simili (ad esempio 2 nodi tg-c), selezionati dai primi nodi con maggiori entries, mostrano un numero di tuple e dei parametri di reliability simili?
5. Per ogni categoria estrarre gli errori e i primi 5 nodi più affetti da essi. Cosa si può notare?

## 5.2.2 Soluzione

### 5.2.2.1 Quesito 1

A partire dal log già filtrato contenente tutti gli errori fatali relativi ai nodi di Mercury, il primo passo è l'uso della Coalescenza temporale per poter raggruppare gli stessi fallimenti segnalati più d'una volta.

Tracciando dunque l'andamento del numero delle tuple in funzione della dimensione della finestra, è stato selezionato per essa il valore di 220 s, trovato in corrispondenza del ginocchio della curva.

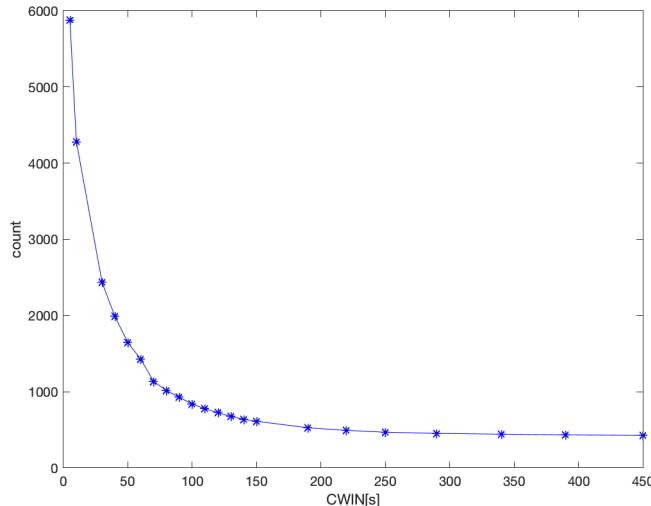


Figura 5.1: Grafico CWIN-tuple

Sulla base di tale valore è stato poi eseguito il tupling, ottenendo così dei campioni per il ttf (interarrivals.txt).

A questo punto è stato possibile a partire da quest'ultimi stimarne la cdf empirica in MATLAB, così come la stessa reliability.

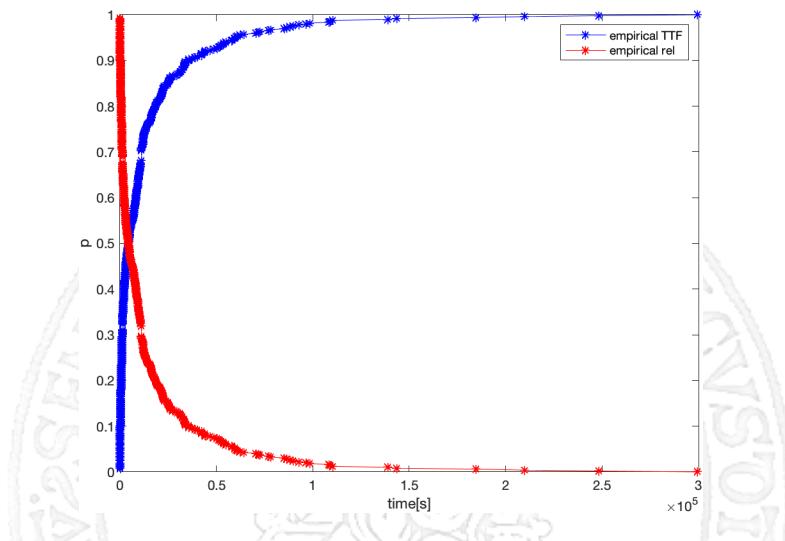


Figura 5.2: Reliability e unreliability empirica

Utilizzando i dati della reliability calcolati è stato possibile eseguire, tramite il tool cftool di matlab, il fitting con 3 diversi modelli : esponenziale, iperesponenziale e weibull.

Si riportano di seguito i risultati.

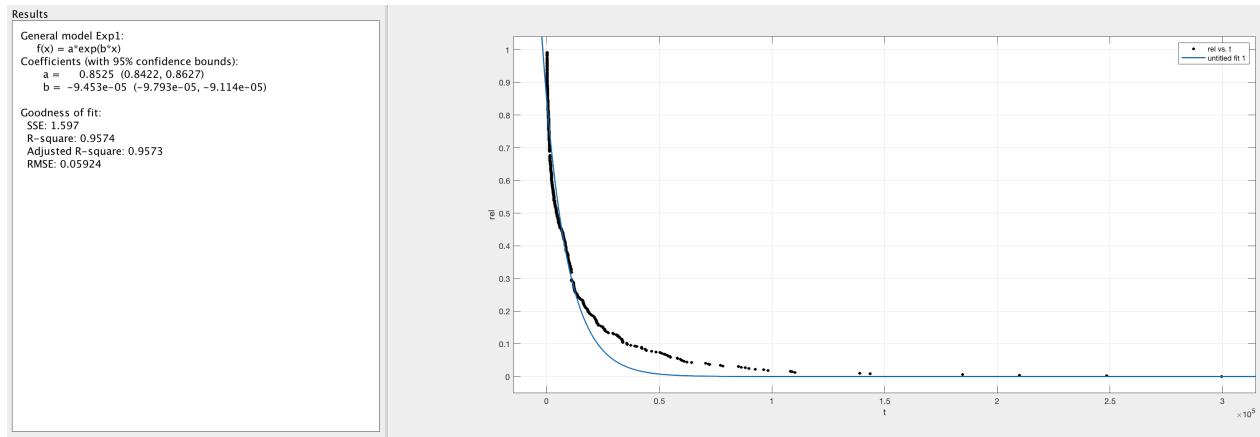


Figura 5.3: Fitting con modello esponenziale

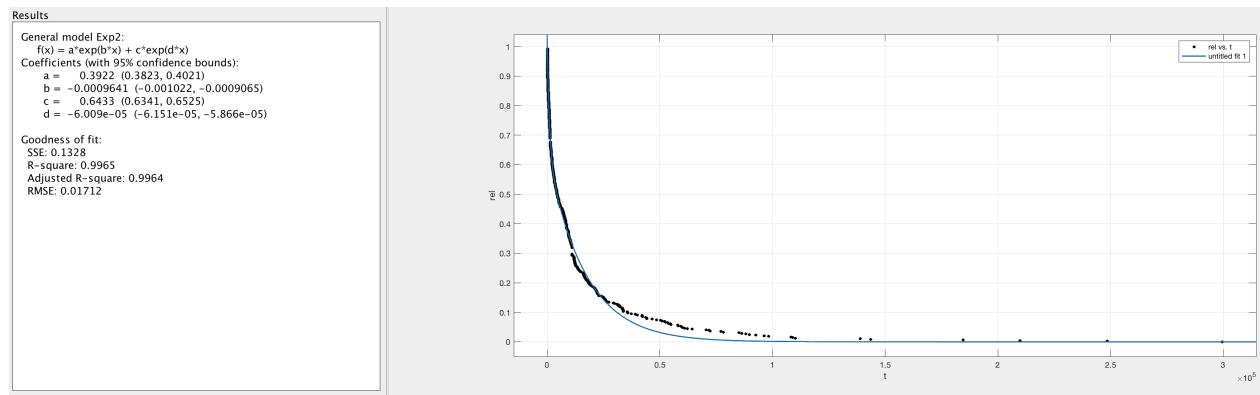


Figura 5.4: Fitting con modello iperesponenziale

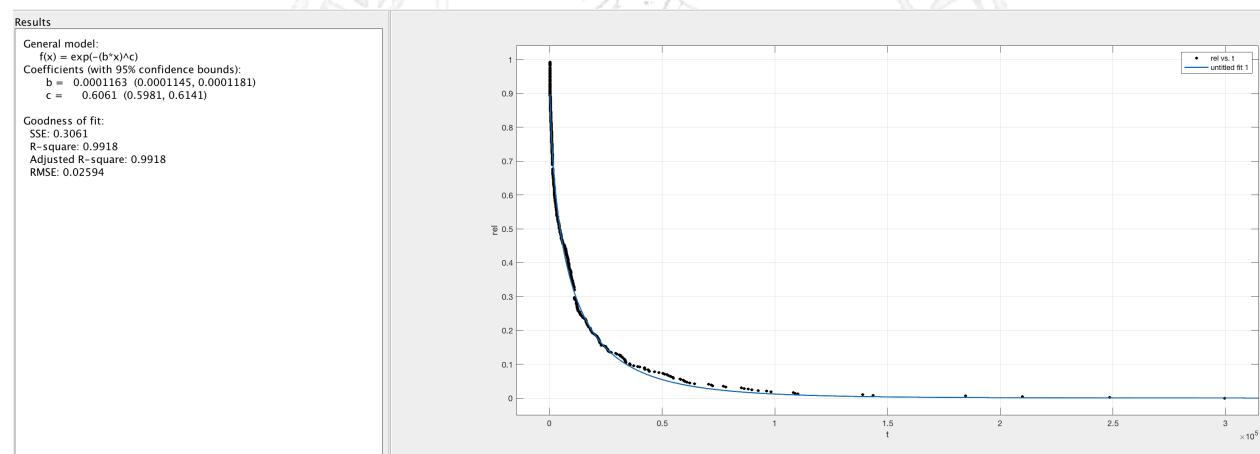


Figura 5.5: Fitting con modello weibull

E' stato utilizzato in matlab il test non parametrico Kolmogorov-Smirnov. L'ipotesi nulla è che i due dataset provengono dalla stessa distribuzione continua.

La decisione per rigettare l'ipotesi nulla è basata sul comparare il p-value con il livello di significatività alfa (che nel nostro caso è il 5%).

Dalla figura si evince che l'ipotesi nulla non è rigettata quando al test si sottopongono le funzioni iperesponenziale e weibull.

```
[H_e,P_e,K_e] = kstest2(rel,exponential_custom(t))
```

```
H_e = logical
```

```
1
```

```
P_e = 6.7992e-05
```

```
K_e = 0.1488
```

```
H_ipe = logical
```

```
0
```

```
P_ipe = 0.7105
```

```
K_ipe = 0.0460
```

```
[H_ipe,P_ipe,K_ipe] = kstest2(rel,iperexp_custom(t))
```

```
H_wei = logical
```

```
0
```

```
P_wei = 0.0678
```

```
K_wei = 0.0853
```

```
[H_wei,P_wei,K_wei] = kstest2(rel, weibull_custom(t))
```

Figura 5.6: Test non parametrico Kolmogorov-Smirnov

### 5.2.2.2 Quesito 2

Di seguito sono riportati i 5 nodi con maggiore numero di entry ottenuti tramite lo script logstatistcs.sh

- tg-c401 62340
- tg-master 4098
- tg-c572 4030
- tg-s044 3224
- tg-c238 1273

In seguito si è deciso di filtrare tramite lo script filter.sh il numero di occorenze per ogni categoria e per ogni nodo ottenendo la tabella.

	DEV	MEM	I/O	NET	PRO	OTH	TOT
tg-c401	50782	11558	0	0	0	0	62340
tg-master	3	1	452	3639	0	3	4098
tg-c572	3176	845	9	0	0	0	4030
tg-s044	0	0	3208	2	0	14	3234
tg-c238	1071	197	2	3	0	0	1273
	55032	12601	3671	3644	0	17	

Tabella 5.1: Occorrenze per ogni nodo e per ogni categoria

Di seguito è mostrato il numero di tuple per ogni nodo al crescere della CWIN.

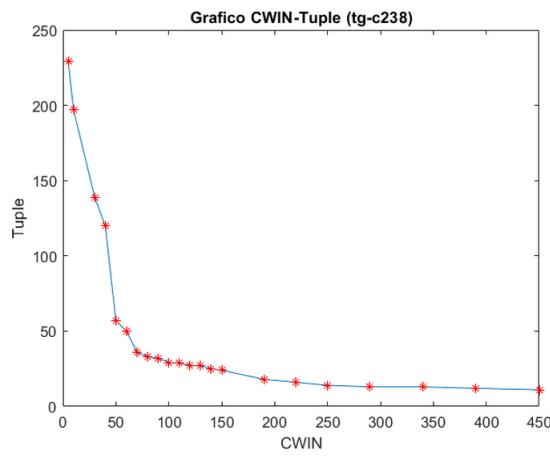


Figura 5.7: Grafico CWIN-tuple tg-c238

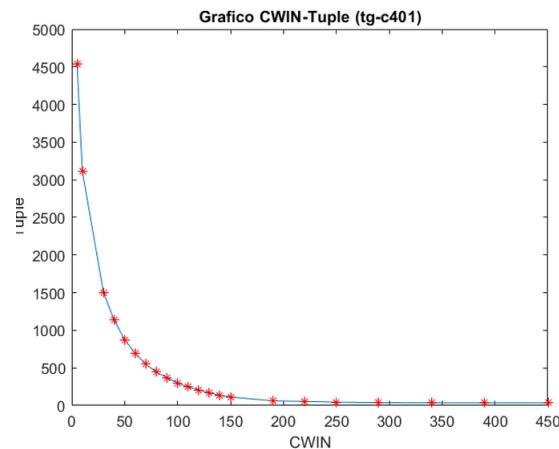


Figura 5.8: Grafico-tuple tg-c401

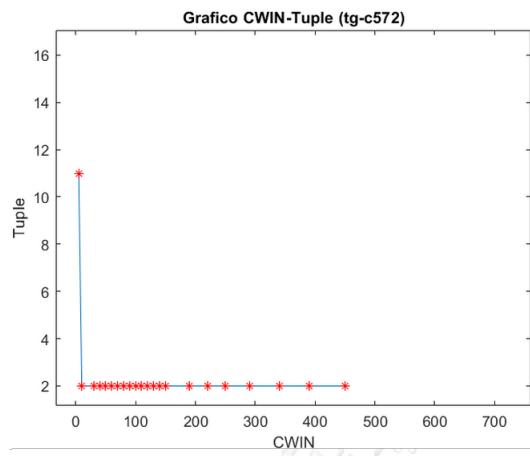


Figura 5.9: Grafico CWIN-tuple tg-c572

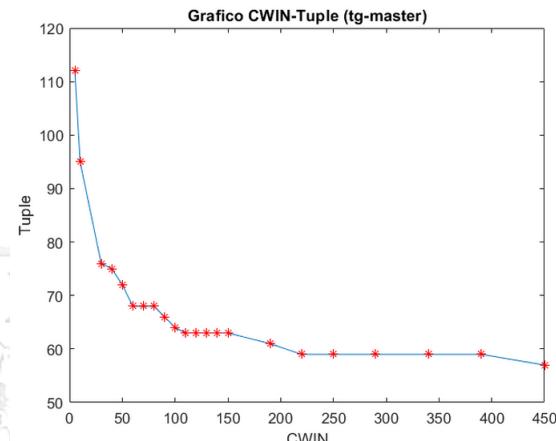


Figura 5.10: Grafico CWIN-tuple tg-master

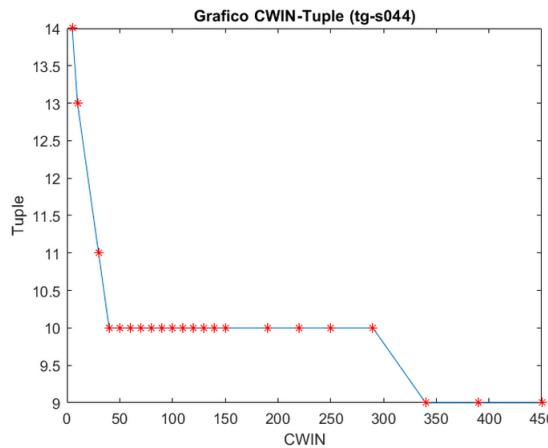


Figura 5.11: Grafico CWIN-tuple tg-s044

La seguente tabella indica, per ognuna delle categorie, la finestra di coalescenza scelta e il numero di tuple relativo.

Nodi	Secondi	Tuple
tg-c401	220	55
tg-master	220	59
tg-c572	10	2
tg-s044	40	10
tg-c238	220	16

Tabella 5.2: Tabella CWIN ottimale per ogni nodo

La CWIN risulta essere simile per i due nodi di computation 401 e 238 e il nodo master.

Inoltre in base al numero di tuple risultanti è possibile constatare che i nodi computation 401 e master fungono da colli di bottiglia.

Successivamente, come da traccia, si è provveduto a plottare le reliability di questi due nodi in quanto hanno un numero di tuple maggiore di 30.

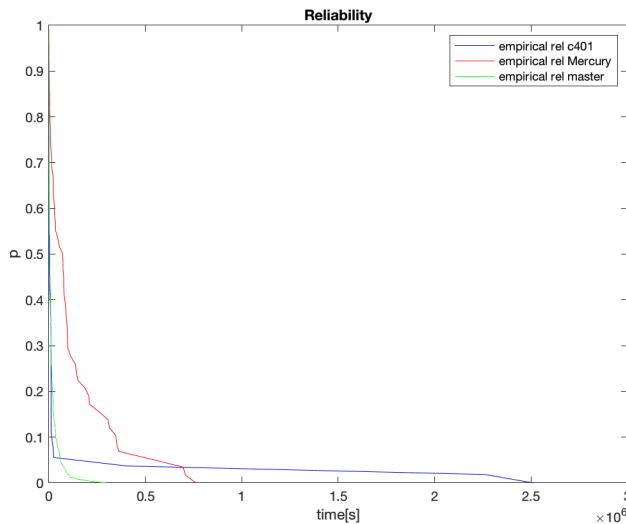


Figura 5.12: Confronto Reliability tra sistema, tg-master e tg-c401

### 5.2.2.3 Quesito 3

Per determinare il corretto CWIN , come per i nodi, si è deciso di plottare il numero di tuple per ogni categoria in funzione della CWIN.

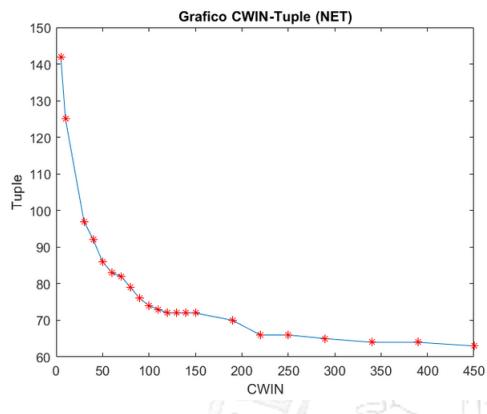


Figura 5.13: 5.20: Grafico CWIN-tuple NET

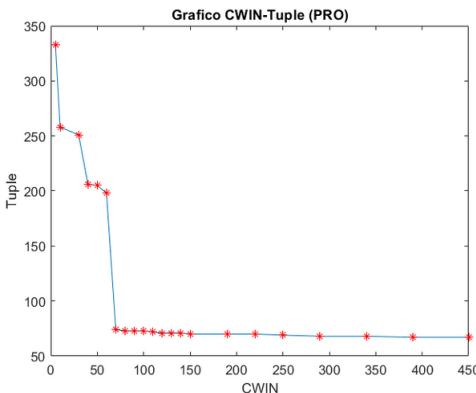


Figura 5.14: Grafico CWIN-tuple PRO

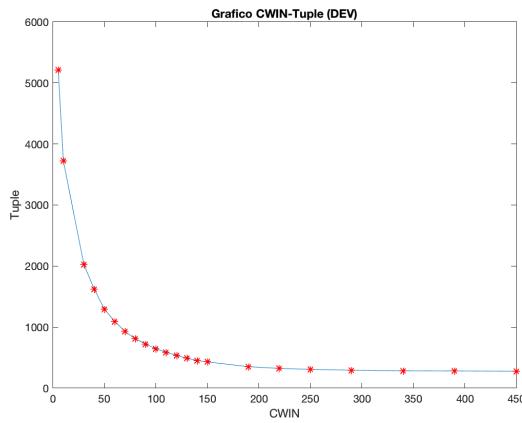


Figura 5.15: Grafico CWIN-tuple DEV

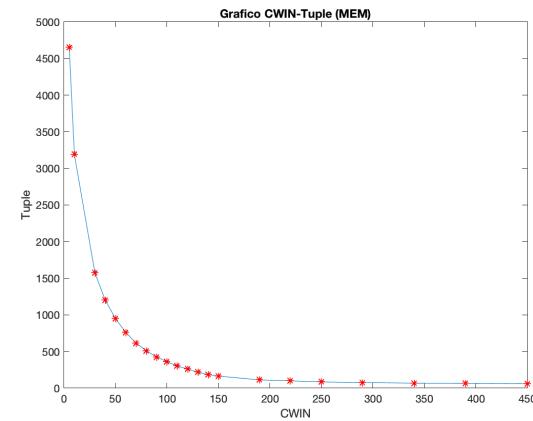


Figura 5.16: Grafico CWIN-tuple MEM

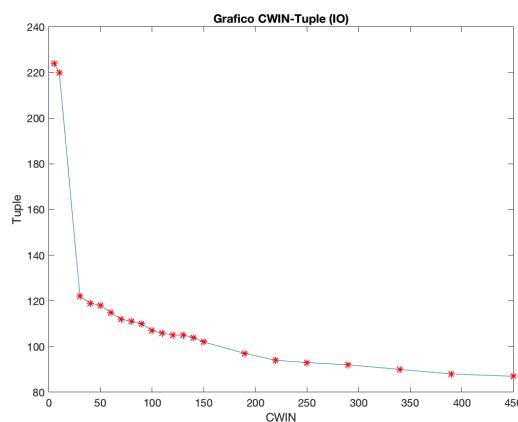


Figura 5.17: Grafico CWIN-tuple IO

Nella seguente tabella sono riportati i numeri di tuple per le CWIN scelte. La categoria con il maggior numero di tuple risulta essere DEV.

Nodi	Secondi	Tuple
NET	220	66
PRO	70	74
MEM	220	100
IO	220	94
DEV	220	323

Tabella 5.3: Tabella CWIN ottimale per ogni categoria

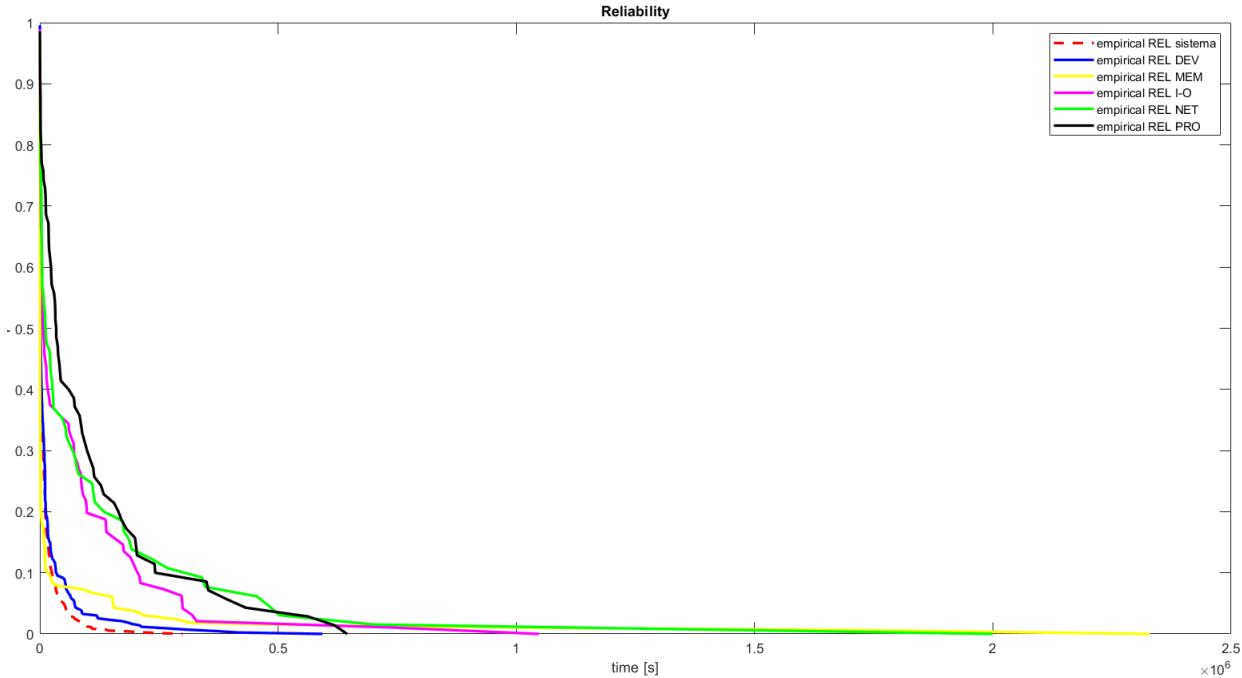


Figura 5.18: Grafico Reliability sistema, DEV, MEM, I-O, NET, PRO

La reliability di MEM è la peggiore fino al punto di breakpoint tra MEM e DEV, successivamente la reliability di DEV risulta la peggiore. Le categorie NET e PRO sono le più reliable.

Analogamente a come fatto in precedenza si andrà a fissare la reliability delle singole categorie con i tre diversi modelli, e per semplicità si riportano di seguito solo i modelli che sono risultati migliori per ognuna. Inoltre per valutare la bontà del fit è stato eseguito il K-S test di cui si riportano i risultati in fig.

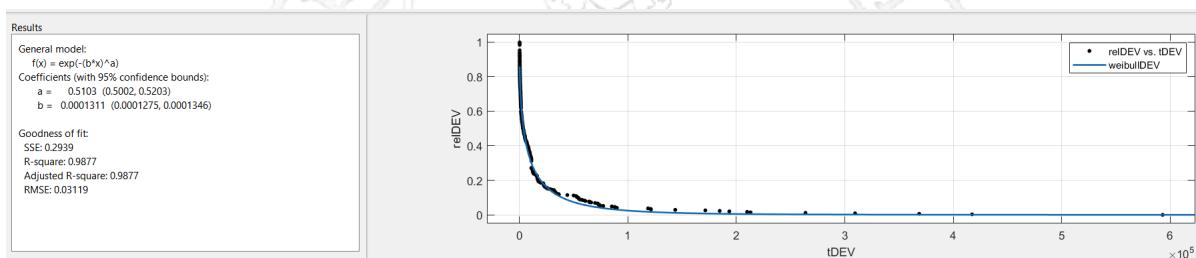


Figura 5.19: Fit DEV weibull

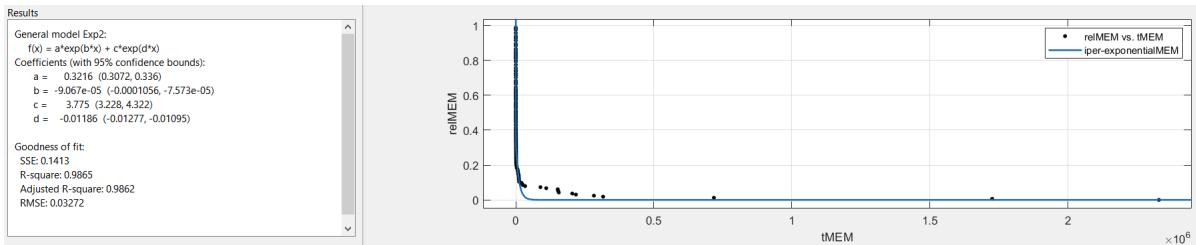


Figura 5.20: Fit MEM iper-esponenziale

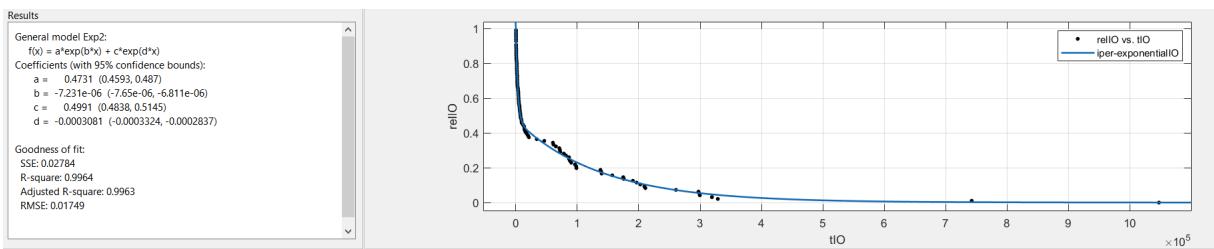


Figura 5.21: Fit I-O iper-esponenziale

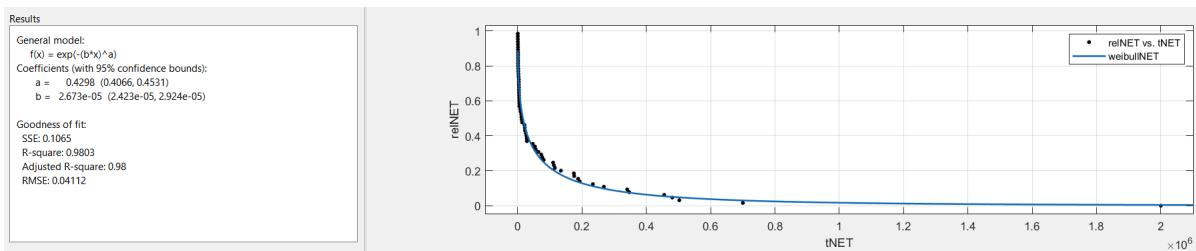


Figura 5.22: Fit NET weibull

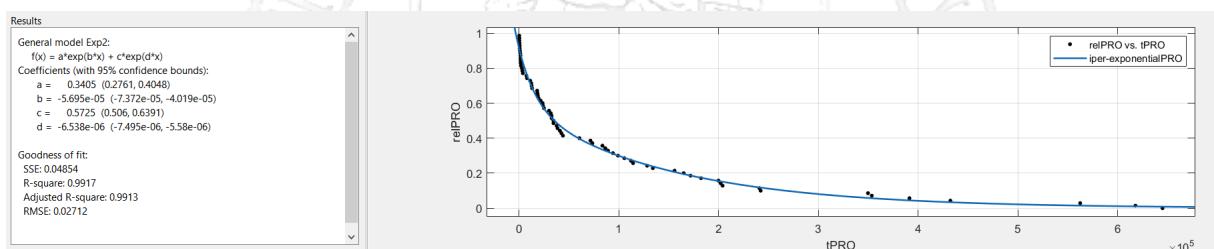


Figura 5.23: Fit PRO iper-esponenziale

```

H_DEV = logical
  0
P_DEV = 0.0793
K_DEV = 0.1020

H_MEM = logical
  0
P_MEM = 0.6451
K_MEM = 0.0882

H_IO = logical
  0
P_IO = 0.9992
K_IO = 0.0526

H_NET = logical
  0
P_NET = 0.8238
K_NET = 0.1077

H_PRO = logical
  0
P_PRO = 0.9505
K_PRO = 0.0857

```

Figura 5.24: risultati K-S test per le distribuzioni delle categorie

#### 5.2.2.4 Quesito 4

I nodi scelti per il confronti sono tg-c401 e tg-c238. Le CWIN per questi due nodi sono state scelte nei punti precedenti e quindi di seguito sarà solo riportato il grafico della reliability.

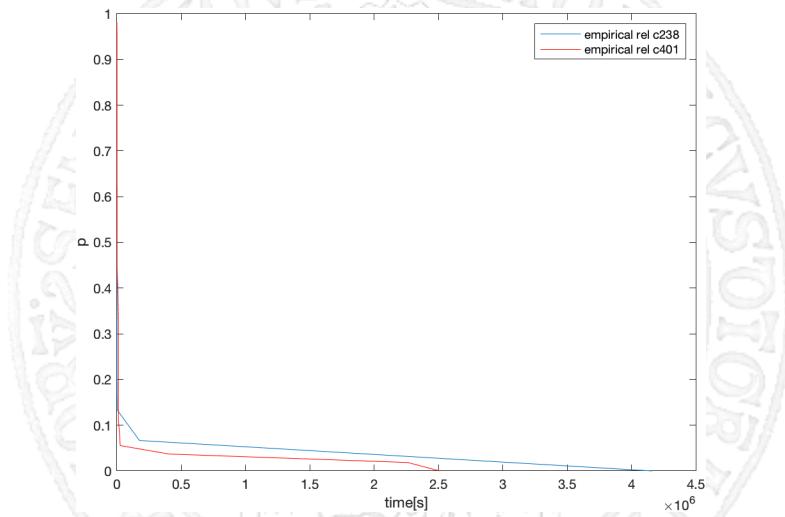


Figura 5.25: Confronto Reliability tg-c238 e tg-c401

La reliability osservata del nodo c238 risulta essere migliore di quella del nodo c401.

### 5.2.2.5 Quesito 5

Per rispondere a questo quesito per ogni categoria sono stati valutati i 5 nodi più influenti in termini di numero di occorrenza.

	ERRORI	NODO 1	NR.	%	NODO 2	NR.	%	NODO 3	NR.	%	NODO 4	NR.	%	NODO 5	NR.	%
DEV	57248	Tg-c401	50782	88.71	Tg-c572	3176	5.55	Tg-c238	1071	1.87	Tg-c242	918	1.6	Tg-c117	263	0.46
MEM	12819	Tg-c401	11558	90.16	Tg-c572	845	6.59	Tg-c238	197	1.54	Tg-c242	149	1.16	Tg-c894	28	0.22
IO	5547	Tg-s044	3208	57.83	Tg-master	452	8.15	Tg-login3	381	6.87	Tg-s038	230	4.15	Tg-c550	222	4
NET	3702	Tg-master	3629	98.3	Tg-c550	33	0.89	Tg-c196	14	0.38	Tg-c238	3	0.08	Tg-s044 - tg-c027	2	0.05
PRO	1504	Tg-c648	616	40.96	Tg-c324	239	15.89	Tg-c284	180	11.97	Tg-c451	159	10.57	Tg-c447	136	9.04

Tabella 5.4: Per ogni categoria i 5 nodi più influenti

Come si evince da essa si ha che:

- per la categoria DEV il nodo maggiormente coinvolto è il tg-c401 con una percentuale di errori pari a 88,71%;
- per la categoria MEM il nodo maggiormente coinvolto è il tg-c401 con una percentuale di errori pari a 90,16%;
- per la categoria I-O il nodo maggiormente coinvolto è il tg-s044 con una percentuale di errori pari a 57,83%;
- per la categoria NET il nodo maggiormente coinvolto è il tg-master con una percentuale di errori pari a 98,3%;
- per la categoria PRO il nodo maggiormente coinvolto è il tg-c648 con una percentuale di errori pari a 40,96%;

Si osservi inoltre

- come gli errori di categoria MEM e DEV siano presenti con frequenza maggiore segli stessi nodi e inoltre come è stato rilevato nelle analisi precedenti tali errori si concentrano principalmente nel nodo di computation 401.
- gli errori di NET sono presenti in netta maggioranza sul nodo tg-manager

## 5.3 FFDA: Blue Gene

Blue Gene/L è un super computer IBM. L'architettura di BG/L è composta da una serie di Rack ognuno dei quali è suddiviso in 2 Midplane.

Ogni Midplane contiene dei nodi e su ognuno di essi sono allocate 16 Compute Card.

Ogni compute Card è formata da 2 SoC da 2 CPU ciascuna (quindi 4 CPU per ogni Compute Card).

Il sistema comunica attraverso specifici nodi installati in ogni Rack: N0, N4, N8 e NC. Su ognuno di essi è installata la I-O Card, che si occupa esclusivamente di operazioni di I-O.

Nel log

- la lettera R seguita da un numero rappresenta la cabina contenente i nodi;
- la lettera M seguita da 00 o 01 rappresenta uno dei Midplane;
- la lettera N seguita dal un numero rappresenta il nodo del midplane;
- la lettera J seguita da un numero compreso tra 1 e 16 identifica la Compute Card;
- La lettera U seguita da un numero da 00 a 11 identifica una delle unità della Compute Card o I/O Card;
- la lettera J seguita da 18, presente solo su alcuni nodi (N0, N4, N8 e NC), rappresenta le I-O Card.

Per quanto riguarda il log del sistema (BGLErrorLog.txt) è costituito dai seguenti campi:

- timestamp
- nodo origine
- card origine
- messaggio di errore

### 5.3.1 Traccia

1. Condurre l'analisi per BGLErrorLog.txt:

- plottare il tuple count e selezionare una CWIN;
- raggruppare le entries utilizzando la CWIN;
- distribuzione della Reliability empirica;
- fit della Reliability empirica (provare tutti i modelli: esponenziale, iper-esponenziale, weibull);
- analizzare i modelli risultanti con il K-S test.

2. Selezionare i primi 5 nodi con maggiori entries; determinare per ogni nodo CWIN e numero di tuple;

- i nodi hanno tutti la stessa CWIN?
- esistono dei colli di bottiglia per la Dependability, cioè dei nodi con un gran numero di tuple rispetto agli altri?
- plottare la Reliability per i nodi con un numero di campioni di interarrivals significativamente grande ( $>30$ )

3. Analizzare le categorie J18-U01 e J18-U11 (numero di tuple e Reliability).

4. Nodi funzionalmente simili (ad esempio 2 nodi I-O), selezionati dai primi nodi con maggiori entries, mostrano un numero di tuple e dei parametri di reliability simili?

5. Quali sono i Rack/nodi con più entries?

## 5.3.2 Soluzione

### 5.3.2.1 Quesito 1

A partire dal log già filtrato contenente tutti gli errori fatali relativi ai nodi di Blue Gene, il primo passo è l'uso della Coalescenza temporale per poter raggruppare gli stessi fallimenti segnalati più d'una volta.

Tracciando dunque l'andamento del numero delle tuple in funzione della dimensione della finestra, è stato selezionato per essa il valore di 220 s, trovato in corrispondenza del ginocchio della curva.

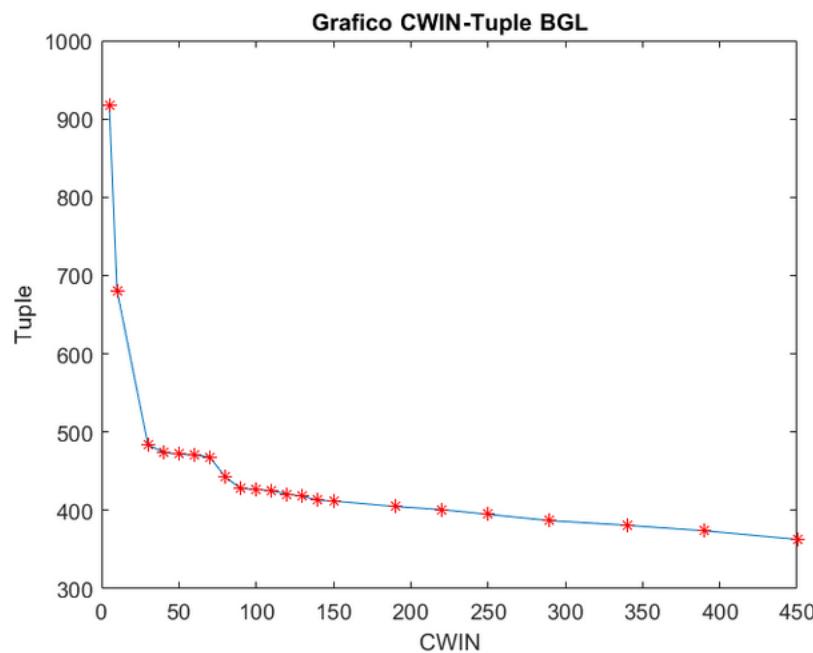


Figura 5.26: Grafico CWIN-tuple

Sulla base di tale valore è stato poi eseguito il tupling, ottenendo così dei campioni per il ttf (interarrivals.txt).

A questo punto è stato possibile a partire da quest'ultimi stimarne la cdf empirica in MATLAB, così come la stessa reliability.

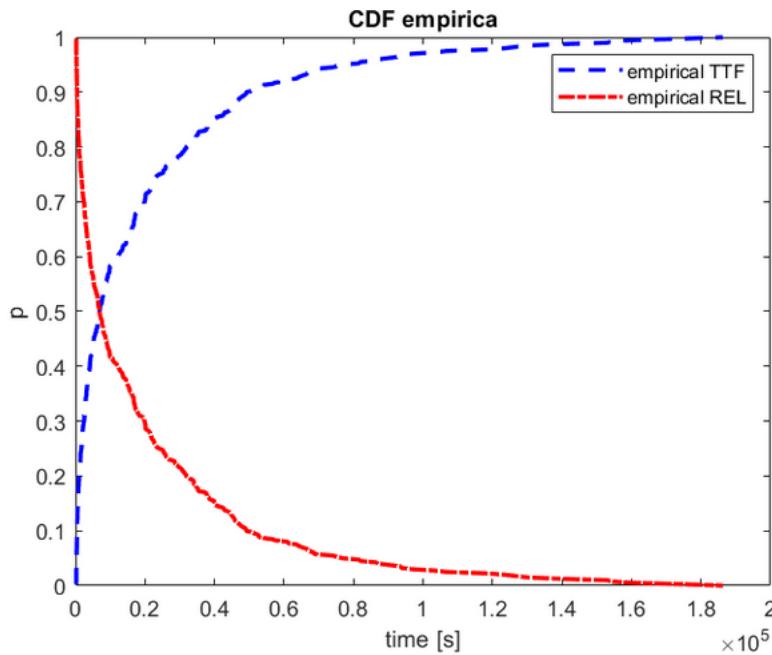


Figura 5.27: Grafico reliability e unreliability

Utilizzando i dati della reliability calcolati è stato possibile eseguire, tramite il tool cftool di matlab, il fitting con 3 diversi modelli : esponenziale, iperesponenziale e weibull.

Si riportano di seguito i risultati.

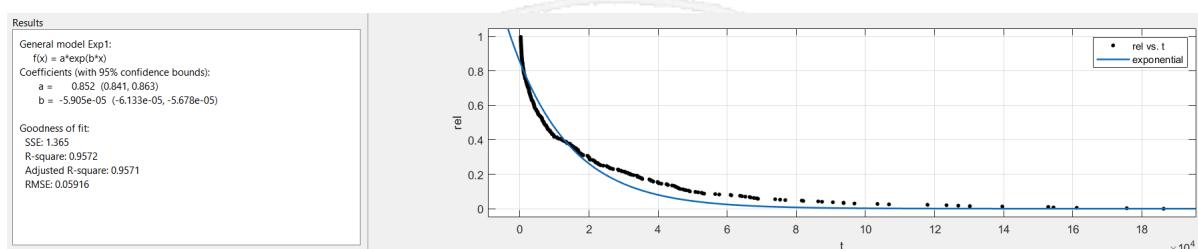


Figura 5.28: Fitting con Modello esponenziale

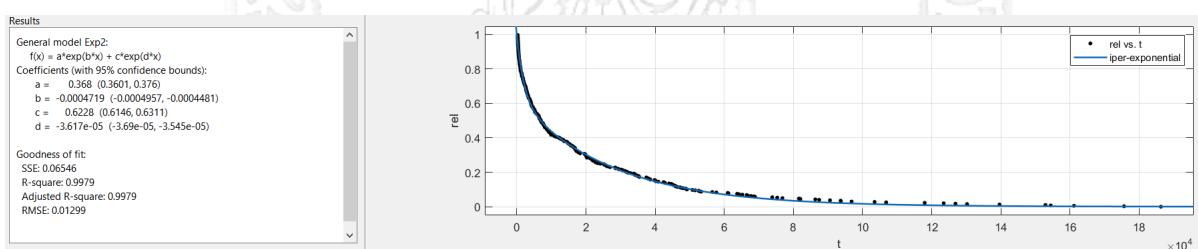


Figura 5.29: Fitting con Modello iper-esponenziale

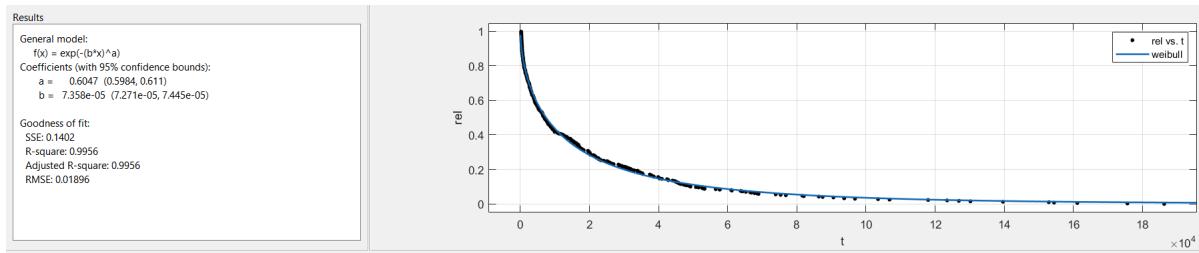


Figura 5.30: Fitting con Modello Weibull

E' stato utilizzato in matlab il test non parametrico Kolmogorov-Smirnov. L'ipotesi nulla è che i due dataset provengono dalla stessa distribuzione continua.

La decisione per rigettare l'ipotesi nulla è basata sul comparare il p-value con il livello di significatività alfa (che nel nostro caso è il 5%).

Dalla figura si evince che l'ipotesi nulla non è rigettata quando al test si sottopongono le funzioni iperesponeziale e weibull.

```
H_exp = logical
1
P_exp = 4.3195e-04
K_exp = 0.1454
```

```
H_iper_exp = logical
0
P_iper_exp = 0.8471
K_iper_exp = 0.0434

H_weibull = logical
0
P_weibull = 0.3000
K_weibull = 0.0689
```

Figura 5.31: Test non parametrico Kolmogorov-Smirnov

### 5.3.2.2 Quesito 2

Di seguito sono riportati i 5 nodi con maggiore numero di entry ottenuti tramite lo script logstatistcs.sh

- R71-M0-N4 1716

- R12-M0-**N0** 1563
- R63-M0-N2 976
- R03-M1-NF 960
- R63-M0-**N0** 791

Di seguito è mostrato il numero di tuple per ogni nodo al crescere della CWIN.

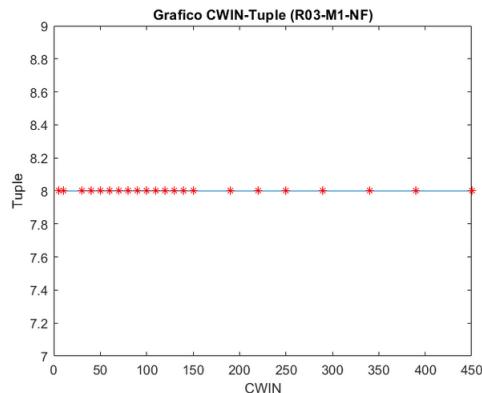


Figura 5.32: Grafico CWIN-tuple R03-M1-NF

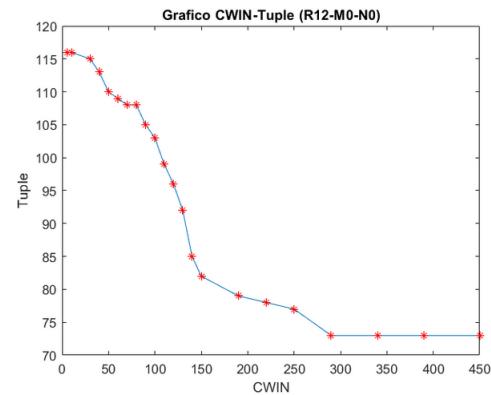


Figura 5.33: Grafico CWIN-tuple R12-M0-N0

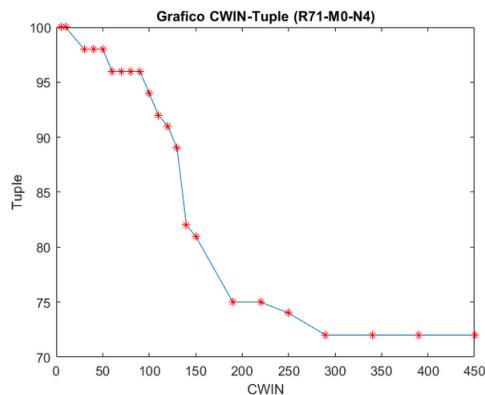


Figura 5.34: Grafico CWIN-tuple R71-M0-N4

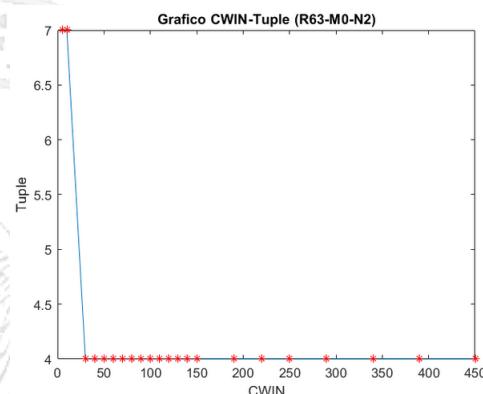


Figura 5.35: Grafico CWIN-tuple R63-M0-N2

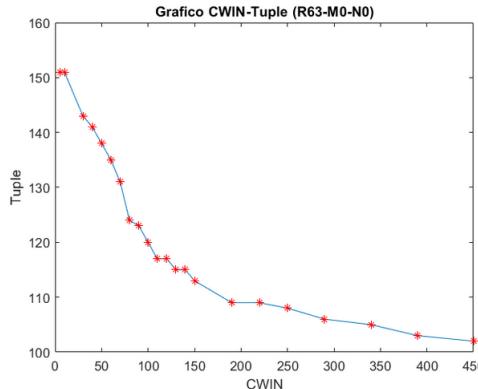


Figura 5.36: Grafico CWIN-tuple R63-M0-N0

La seguente tabella indica, per ognuna delle categorie, la finestra di coalescenza scelta e il numero di tuple relativo.

Nodi	Secondi	Tuple
R03-M1-NF	5	8
R12-M0-N0	290	73
R71-M0-N4	290	72
R63-M0-N2	30	4
R63-M0-N0	390	103

Tabella 5.5: Tabella CWIN-tuple

La CWIN e il numero di tuple risulta pressoché lo stesso per il nodo R12-M0-N0 e il nodo R71-M0-N4.

Inoltre in base al numero di tuple risultanti è possibile constatare che i nodi R12-M0-N0, R71-M0-N4 e R63-M0-N0 hanno un numero di fallimenti che è un ordine di grandezza maggiore rispetto agli altri e quindi fungono da colli di bottiglia.

Successivamente, come da traccia, si è provveduto a plottare le reliability di questi tre nodi in quanto hanno un numero di tuple maggiore di 30.

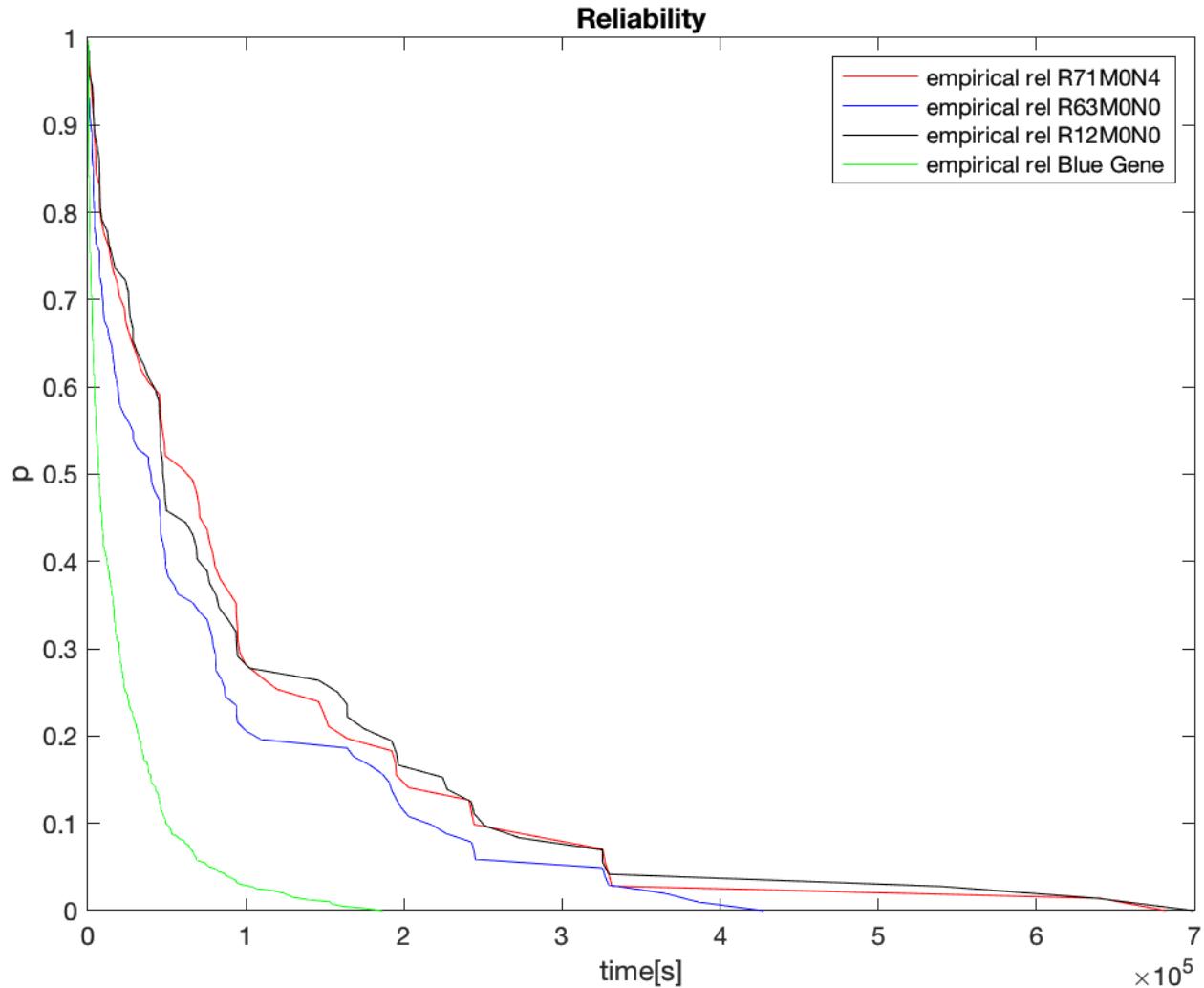


Figura 5.37: Confronto Reliability sistema, R71-M0-N4, R63-M0-N0 e R12-M0-N0

### 5.3.2.3 Quesito 3

Per determinare il corretto CWIN delle due categorie **J18-U01** e **J18-U11**, come per i nodi, si è deciso di plottare il numero di tuple per ogni categoria in funzione della CWIN.

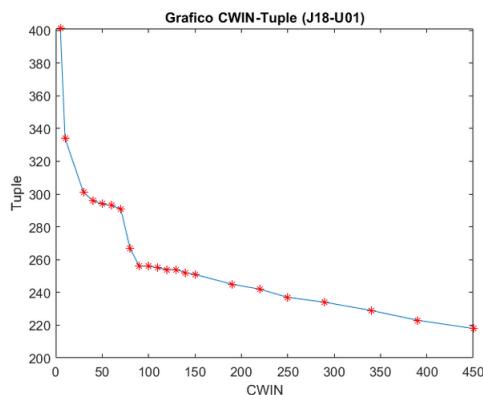


Figura 5.38: Grafico CWIN-tuple J18-U01

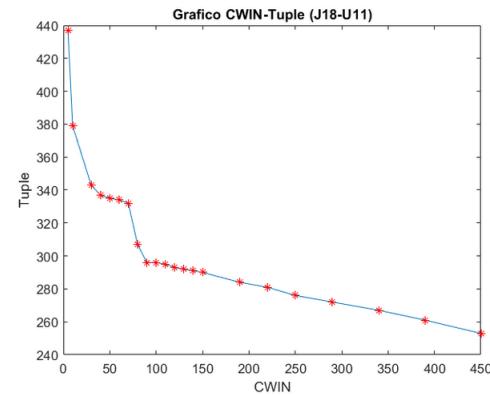


Figura 5.39: Grafico CWIN-tuple J18-U11

Nella seguente tabella sono riportati i numeri di tuple per le CWIN scelte.

Nodi	Secondi	Tuple
J18-U01	220	241
J18-U11	190	284

Tabella 5.6: Tabella CWIN ottimale per ogni categoria

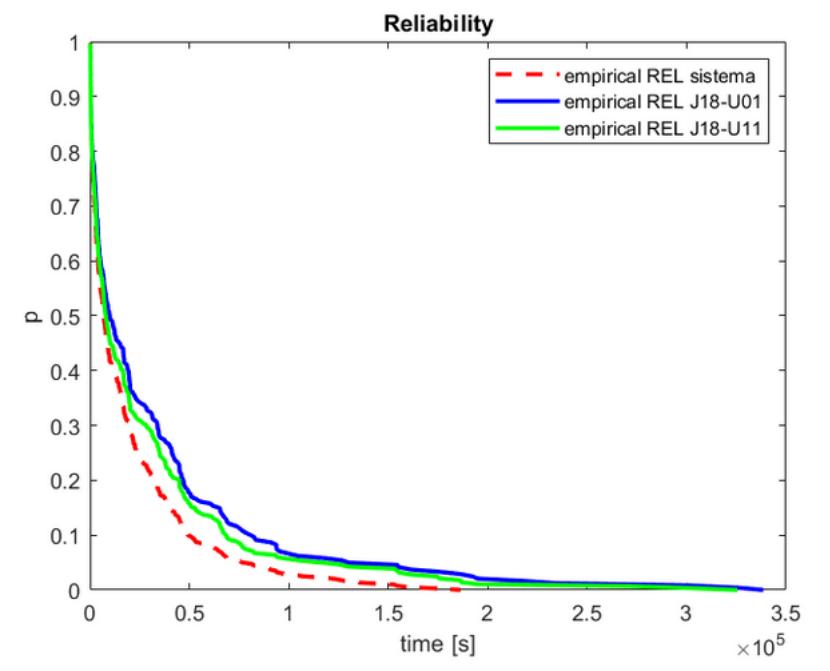


Figura 5.40: Grafico Reliability sistema, J18-U01 e J18-U11

Come si può notare dal graco la categoria J18-U01 risulta essere più Reliable della categoria J18-U11.

### 5.3.2.4 Quesito 4

Tra i primi 10 nodi individuato dallo script LogStatistics.sh, quelli scelti per il confronti sono R62-M0-N4 e R63-M0-N4.

Le dimensioni delle CWIN scelte sono: 290 con 90 tuple per il nodo R62-M0-N4; 220 con 109 tuple per il nodo R63-M0-N4. Di seguito si plottano le Reliability dei due nodi.

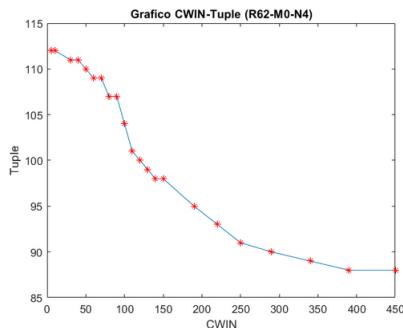


Figura 5.41: Grafico CWIN-tuple R62-M0-N4

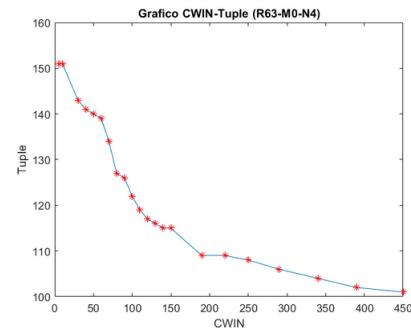


Figura 5.42: Grafico CWIN-tuple R63-M0-N4

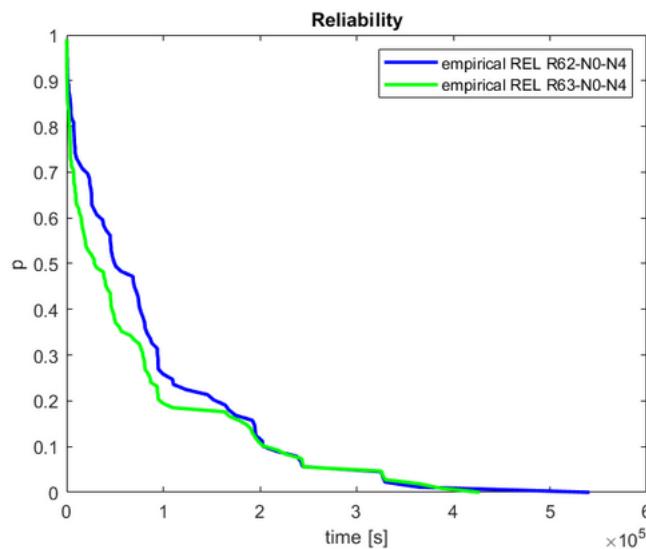


Figura 5.43: Confronto Reliability R62-M0-N4 e R63-M0-N4

Dal grafico si può intuire che le due Reliability risultano essere molto simili tra loro. Si può quindi affermare che i parametri delle Reliability sono molto simili tra loro.

### 5.3.2.5 Quesito 5

```
= Total error entries =
125624
= Backup by CATEGORY =
J18-U11 50055
J18-U01 49932
J14-U01 2257
J12-U01 1877
J07-U01 1780
J10-U11 1333
J03-U11 1020
J16-U11 973
J06-U11 960
J11-U11 888 Network
J17-U11 824
J09-U01 808
J16-U01 753
J09-U11 746
J08-U01 741
J03-U01 701
J11-U01 700
J05-U11 670
J04-U01 655
J13-U01 647
J15-U01 633
J17-U01 602
J12-U11 596
J04-U11 593
J13-U11 563
J08-U11 560
J10-U01 554
J02-U01 524
J05-U01 456
J15-U11 454
J02-U11 449
J14-U11 445
J07-U11 445
J06-U01 430

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

Figura 5.44: Statistiche BG/L

Rack	Event Logs
R63	7192
R62	4057
R57	3581
R56	3527
R46	3485
R03	2982
R12	2976
R71	2815
R61	2366
R63	2230

Figura 5.45: Rack con più errori

# Capitolo 6

## Cloud

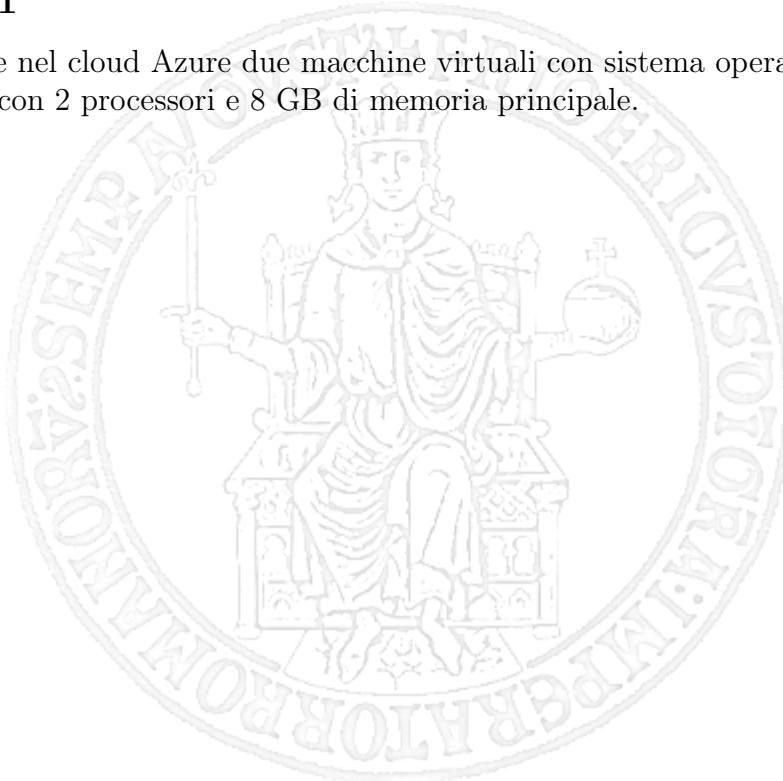
### 6.1 Cloud

#### 6.1.1 Traccia

- Crea un gruppo di virtual machine su una piattaforma di cloud computing per esempio Azure.
- Crea un load balancer ed effettua richieste da un client o un gruppo di client e fai delle considerazioni sulle performance

#### 6.1.2 quesito 1

Sono state istanziate nel cloud Azure due macchine virtuali con sistema operativo Ubuntu Server 18.04 LTS ciascuna con 2 processori e 8 GB di memoria principale.



### Create a virtual machine

[Basics](#) Disks Networking Management Advanced Tags Review + create

Create a virtual machine that runs Linux or Windows. Select an image from Azure marketplace or use your own customized image.

Complete the Basics tab then Review + create to provision a virtual machine with default parameters or review each tab for full customization.

#### Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \* ⓘ

Free Trial

Resource group \* ⓘ

(New) ES\_IMPIANTI

[Create new](#)

#### Instance details

Virtual machine name \* ⓘ

SERVER1

Region \* ⓘ

(US) East US

Availability options ⓘ

Availability set

Availability set \* ⓘ

(new) SERVERGROUP

[Create new](#)

Image \* ⓘ

Ubuntu Server 18.04 LTS

[Browse all public and private images](#)

Azure Spot instance ⓘ

Yes  No

Size \* ⓘ

**Standard D2s v3**

2 vcpus, 8 GiB memory (€59.10/month)

[Change size](#)

Figura 6.1: Settings delle due macchine virtuali che fungono da Server

### 6.1.3 quesito 2

Di seguito sono mostrati due grafici in cui sono mostrate le richieste in entrata e in uscita dei due server sottoposti a uno stesso workload (Random(Httprequest:10MB,1MB,100KB); Costant TP timer 250 richieste al minuto per thread; 100 thread) ma con politiche di Load balancer differenti.

Nel primo grafico la politica del load balancer è quella sempre a livello rete/trasporto ma di tipo hash: il carico si distribuisce quasi equamente sui due server.

Nel secondo grafico la politica di smistamento del load balancer è quella IP: essendo tutti i pacchetti inviati dallo stesso ip sorgente le richieste risultano essere prese in carico da un solo server (in questo caso il server 1).

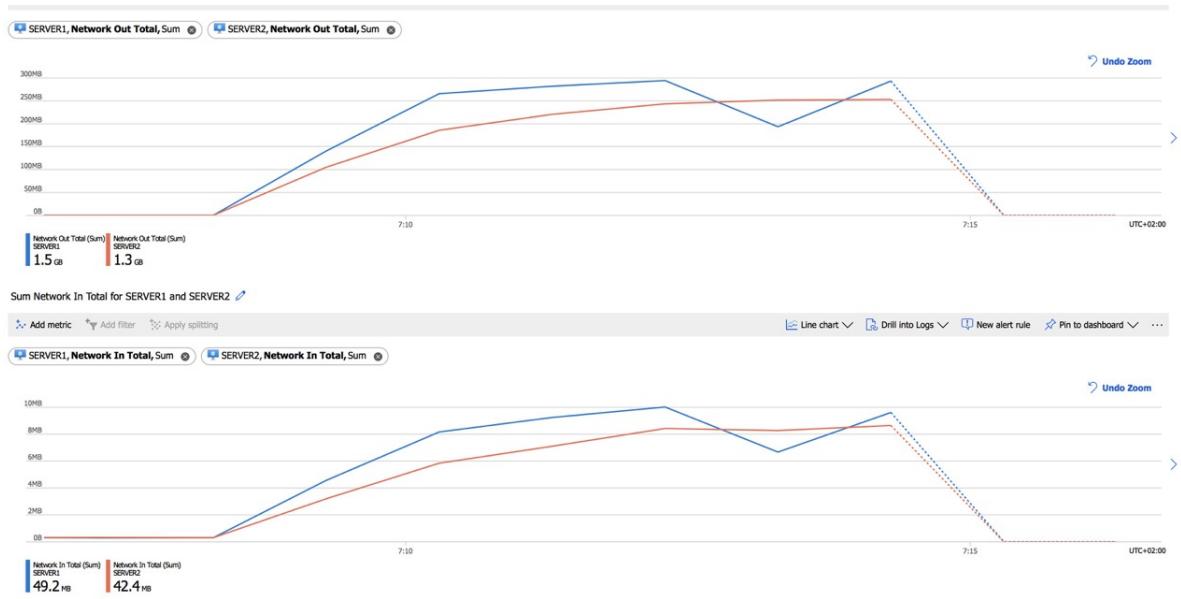


Figura 6.2: Network Out/In per i due Server con Load Balancer con politica Hash



Figura 6.3: Network Out/In per i due Server con Load Balancer con politica Client IP