

Impianti di Elaborazione

Homework

Francesco Iannaccone M63001324, Matteo Conti M63001317

2022-2023

Indice

1 Benchmark	3
1.1 Raccolta dei dati	4
1.2 Analisi dei dati	6
2 Web Server	9
2.1 Capacity Test	9
2.1.1 Indice di equità	13
2.2 Workload Characterization	14
2.3 Design of Experiment	22
3 Workload Characterization	28
3.1 Principal Component Analysis	33
3.2 Clustering	34
4 Regressione	37
4.1 Esercizio 1 - Mail server	37
4.2 Esercizio 2 - Mail Server	41
4.3 Esercizio 3 - VmStat	44
4.4 Esercizio 4 - Predizione	57
5 Reliability	63
5.1 Esercizio 1 - RBD	63
5.2 Esercizio 2 - Confronto tra sistemi	65
5.3 Esercizio 3 - Skip ring	68

5.4	Esercizio 4 - RBD	71
5.5	Esercizio 5 - Reliability sistema di pilotaggio	75
6	FFDA	80
6.1	Analisi dei log di Mercury	80
6.2	Analisi dei log di Blue Gene	87
6.3	Confronto della reliability dei due sistemi	91
6.4	Altre analisi	92
6.4.1	Scelta della finestra di coalescenza	92
6.4.2	Reliability bottlenecks	96
6.4.3	Correlazione tra nodi e tipi di errore	98
7	Teoria delle code	101
7.1	Sistema in analisi	101
7.2	Metodologia e risultati ottenuti	102

Capitolo 1

Benchmark

Per questo primo esercizio è richiesto di effettuare la comparazione di due PC differenti utilizzando come benchmark il programma N_Body. Questo simula l'evoluzione di un sistema di N corpi che sono sotto l'influenza della forza gravitazionale.

Il System Under Test di questo esercizio sarà quindi il processore e il Component Under Study la floating-point unit, visto che il benchmark stressa questo specifico componente dell'ALU.

L'obiettivo è quello di dimostrare che i tempi di esecuzione sui due processori risultino statisticamente differenti, in quanto appartenenti a due generazioni diverse della famiglia Intel. Inoltre, la capacità di RAM nei due sistemi sistema è differente mentre invece il sistema operativo sottostante è lo stesso.

La macchina di Francesco ha le seguenti caratteristiche:

- Processore: Intel Core i7 1065G7
- Memoria RAM: 16 GB
- Sistema Operativo: Windows 10

La macchina di Matteo presenta invece le seguenti specifiche:

- Processore: Intel Core i5 8250U @ 1.60GHz

- Memoria RAM: 8,00GB Dual-Channel DDR4 @ 1064MHz
- Sistema Operativo: Windows 10

1.1 Raccolta dei dati

La prima fase di questa analisi è quella di raccolta dei dati tramite delle operazioni di misurazione diretta. La metrica utilizzata è il tempo di esecuzione della simulazione. La prima condizione fondamentale da garantire è quella di indipendenza tra i campioni raccolti. L'indipendenza dei campioni è una condizione fondamentale per la nostra analisi, si tratta infatti di una delle condizioni che ci consentiranno di applicare il teorema del limite centrale. Allo scopo di garantire l'indipendenza tra i campioni raccolti tra ogni misurazione le macchine sotto analisi sono state riavviate, in modo da mitigare gli effetti di aumento delle performance per merito della cache.

Il numero di corpi da simulare è stato scelto pari a 50000, per ogni esperimento sono inoltre raccolte 5 misurazioni, delle quali viene poi calcolata la media campionaria. Sono stati in totale effettuati 40 esperimenti raccogliendo altrettanti campioni. In totale, quindi, lo script dell'n-body è stato eseguito per 200 volte su entrambe le macchine.

In tal modo sono state verificate le ipotesi del teorema del limite centrale, il quale afferma che, raccolti $n \geq 30$ campioni indipendenti da una popolazione, la distribuzione delle medie di tali campioni, al crescere del numero di osservazioni indipendenti, tende ad una distribuzione normale di media μ e varianza $\frac{\sigma}{\sqrt{n}}$. Nelle analisi successive tale risultato permetterà l'applicazione di test parametrici, che assumono la normalità dei dati in input.

PC1	PC2
13669,8	22657,2
26864	22549,8
24427	19561,2
30184,2	11234,5
18717,4	26548,3
37073,4	20871,8
13899,4	14611,8
22899,4	21550,2
20587,8	14613,6
19749,6	19214,5
13855,2	13698,2
15770	10745,1
22243,8	21233,4
22502	14764,6
28259,2	22478,9
33273,8	32169,6
18671	16981,2
22939,8	26489,2
21387	14613,8
25219	24612,2
22347,6	13620,4
20588	18784,4
29118,6	16621,8
35787,4	24146,8
25456,4	11622,2
21106,8	14620,6
23320,4	16627,2
25322,4	22336,8
20442,2	22971,2
19830	6630,6

Figura 1.1: Risultati sperimentali ottenuti.

Abbiamo calcolato, inoltre, il numero minimo di campioni indipendenti al fine di approssimare media e varianza del campione con quelli della popolazione. Dato un errore E del 10% e un intervallo di confidenza del 95%, si utilizza la seguente formula:

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

Si ha quindi:

$$n_1 = \left(\frac{1.96 \cdot 5378}{2261} \right)^2 \approx 22$$

$$n_2 = \left(\frac{1.96 \cdot 6954}{1998} \right)^2 \approx 38$$

I risultati ottenuti hanno evidenziato una dimensione campionaria uguale o inferiore al numero dei campioni già raccolti, di conseguenza si è ritenuto superfluo raccogliere osservazioni aggiuntive.

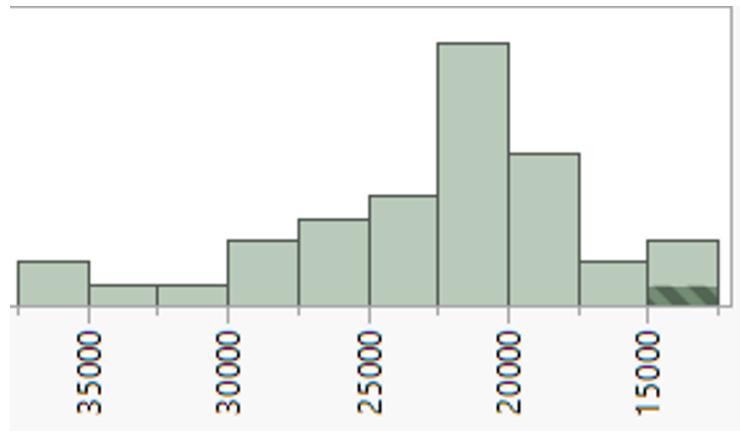


Figura 1.2: Distribuzione delle medie campionarie per il primo PC

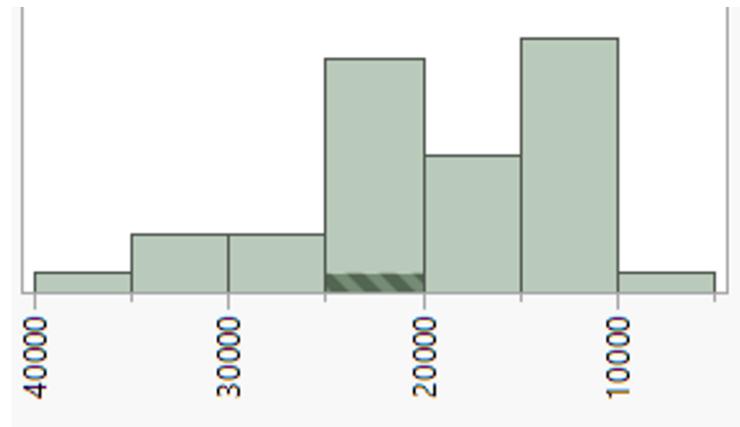


Figura 1.3: Distribuzione delle medie campionarie per il secondo PC

Non si è ritenuto opportuno effettuare analisi per quanto riguarda la distribuzione dei dati poiché il teorema del limite centrale ci assicura la normalità delle distribuzioni.

1.2 Analisi dei dati

Il confronto dei due sistemi richiede un'attenta analisi statistica dei risultati sperimentali ottenuti per i tempi di esecuzione. In prima analisi, fissato il livello di confidenza del 95%, abbiamo provveduto al calcolo degli intervalli di confidenza contenenti la media delle popolazioni originarie dei tempi di

esecuzione. Questi possono essere calcolati con la seguente formula:

$$\hat{X} - \frac{\sigma * z_{\frac{\alpha}{2}}}{\sqrt{n}} \leq \mu \leq \hat{X} + \frac{\sigma * z_{\frac{\alpha}{2}}}{\sqrt{n}}$$

Per il calcolo dell'intervallo, considerando l'elevato numero di campioni, è stata utilizzata la deviazione standard campionaria al posto di quella della popolazione, operando l'approssimazione $S \simeq \sigma$, valida per una dimensione campionaria $n \geq 40$.

Utilizzando JMP otteniamo quindi il seguente risultato:

Intervalli di confidenza				
Parametro	Stima	Cl inferiore	Cl superiore	1-Alfa
Media	22610,72	20890,66	24330,77	0,950
Dev std	5378,281	4405,678	6905,904	0,950

Intervalli di confidenza				
Parametro	Stima	Cl inferiore	Cl superiore	1-Alfa
Media	19982,16	17852,79	22111,54	0,950
Dev std	6658,138	5454,087	8549,286	0,950

Figura 1.4: Intervalli di confidenza, a sinistra il risultato della misura per il computer con il processore i5, a destra quello i7.

Il valore dei tempi di esecuzione osservato sperimentalmente è quindi pari a:

$$T_1 = 22610.72 \pm 1720.055ms$$

$$T_2 = 19982.16 \pm 2129.375ms$$

Come è possibile osservare dalla figura 1.4 e dai risultati ottenuti, gli intervalli di confidenza si sovrappongono ma la media di uno non è nell'intervallo dell'altro. Di conseguenza il test visivo non è conclusivo ed è, invece, necessario eseguire il t-test per verificare l'eventuale appartenenza dei due campioni alla stessa popolazione d'origine.

Dal momento che non esiste una corrispondenza uno a uno tra le coppie di campioni e i campioni prelevati sono indipendenti, è stato scelto di effettuare un unpaired t-test.

Inizialmente, è stata quindi verificata l'uguaglianza tra le varianze delle due distribuzioni: come si può analizzare dalla figura 1.5, il box plot a destra è molto più alto rispetto a quello a sinistra, il che permette di affermare che le varianze non sono uguali.

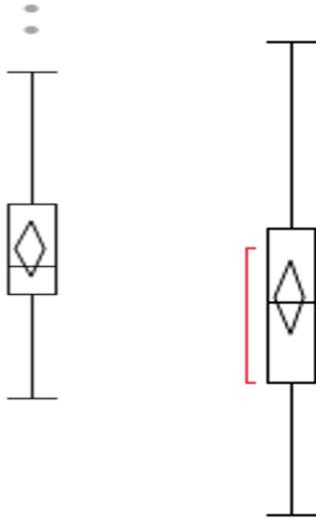


Figura 1.5: Bot plot associato ai due gruppi di campioni

E' stato quindi applicato un Two-sample t-test tramite uno script Python, senza assumere che le varianze delle popolazioni siano uguali. In questo test, l'ipotesi nulla è che i due vettori in input provengano da popolazioni con la stessa media. Il risultato, visibile in figura 1.6, è un p-value basso, che indica che l'evidenza empirica è fortemente contraria all'ipotesi nulla che quindi va rigettata.

Otteniamo quindi che i campioni misurati sono statisticamente differenti e che provengono da due distribuzioni diverse. I tempi di esecuzione ottenuti eseguendo benchmark sul computer con il processore i5 risultano essere nel complesso significativamente più lunghi di quelli di quelli ottenuti su quello con il processore i7. I risultati ottenuti risultano, in conclusione, in linea a quanto previsto all'inizio del paragrafo, la seconda macchina risulta infatti più veloce della prima.

```
PS C:\Users\itama\Desktop\Impianti di Elaborazione\Homework\Benchmark> & C:/Python310/python.exe c:/Users/itama/Desktop/ttest2.py
Il p-value ottenuto in output è uguale a 0.04619784186843966
PS C:\Users\itama\Desktop\Impianti di Elaborazione\Homework\Benchmark>
```

Figura 1.6: Risultato two-sample t-test

Capitolo 2

Web Server

2.1 Capacity Test

Il test di capacità è un test molto utilizzato nell'ambito delle Performance Analysis in quanto permette di trovare il punto di funzionamento ideale del SUT (System Under Test), ovvero il sistema che si intende testare attraverso uno stimolo esterno.

In questo homework il SUT è rappresentato da un Web Server Apache2 in esecuzione su una macchina virtuale con sistema operativo Linux (distribuzione Ubuntu 20-04).

Le metriche che saranno utilizzate per caratterizzare le performance del web server sono le seguenti:

- response time: intervallo di tempo che intercorre tra la richiesta di una risorsa e il completamento della risposta da parte del server;
- throughput: numero di richieste per unità di tempo che il server è in grado di soddisfare.

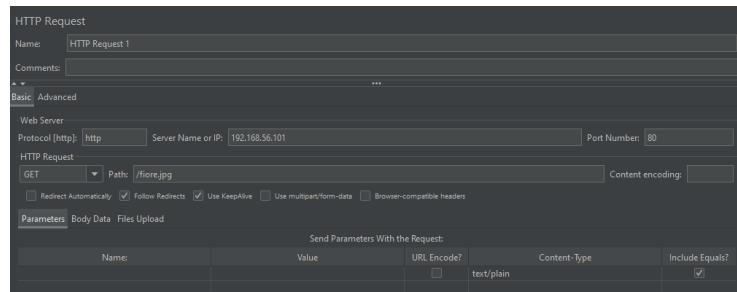
Per effettuare il capacity test è stato configurato un Test Plan tramite il tool Apache JMeter. L'elemento principale di un test plan è il thread group, ovvero un insieme di utenti virtuali che effettuano delle richieste verso il server.

I parametri del thread group sono stati settati nel seguente modo:

- Numero di thread: 50;
- Ramp-up period: 1 s;
- Loop count: infinito;
- Duration: 300 s.

Successivamente, sono stati aggiunti dei sampler di tipo HTTP request, uno per ogni risorsa memorizzata nel Web Server, in particolare nel path /var/www/html. Per le richieste HTTP sono stati specificati:

- Nome della richiesta;
- protocollo, IP e porta su cui contattare il server: in questo caso il protocollo sarà sempre HTTP, l'IP sarà quello assegnato da VirtualBox e la porta 80 standard di HTTP;
- il tipo di richiesta HTTP: saranno sempre richieste con metodo GET;
- il path della risorsa: è stato inserito unicamente il nome della risorsa perchè tutte le risorse sono state memorizzate nel path di default.



Per quest'esercizio, sono state create 12 risorse differenti, con dimensioni variabili da 1KB fino ad arrivare a 16MB.

Le richieste HTTP sono gestite da un Random Controller, che permette ad un thread di fare una richiesta in modo random, al fine di rendere il piano di test quanto più realistico possibile. Infine, nel thread group, oltre al Random Controller, è stato aggiunto un listener di tipo Summary Report, che ha

permesso il collezionamento del parametro di throughput e di salvare il report in un file .csv, e un Constant Throughput Timer (CTT), che aveva lo scopo di settare un throughput target da raggiungere per tutti i thread attivi nel thread group corrente. In particolare, il valore del CCT è stato fatto variare durante l'analisi da 500 fino a 3500.

I dati sono stati raccolti effettuando 3 osservazioni indipendenti per ogni valore di CTT e tra un'osservazione e un'altra la macchina virtuale che ospitava il Web Server veniva riavviata.

Di conseguenza, i valori di response time sono stati calcolati facendo la mediana della colonna Elapsed dei 3 report per un dato valore di CTT e successivamente è stata effettuata la media tra i 3 valori rappresentanti le 3 osservazioni. La scelta di calcolare la media o la mediana dei dati è dettata dal coefficiente di variazione, che aiuta a quantificare in modo oggettivo quanto è grande il valore della deviazione standard rispetto alla media, e quindi il livello di dispersione dei dati rispetto a questa. Nel nostro caso è stata selezionata la media poiché il COV è risultato per tutti i casi molto ridotto ed inferiore a 5.

Invece, il throughput è stato raccolto direttamente dal summary report e quindi è bastato calcolare la media tra le varie ripetizioni per ottenere un unico valore per ogni valore di CTT.

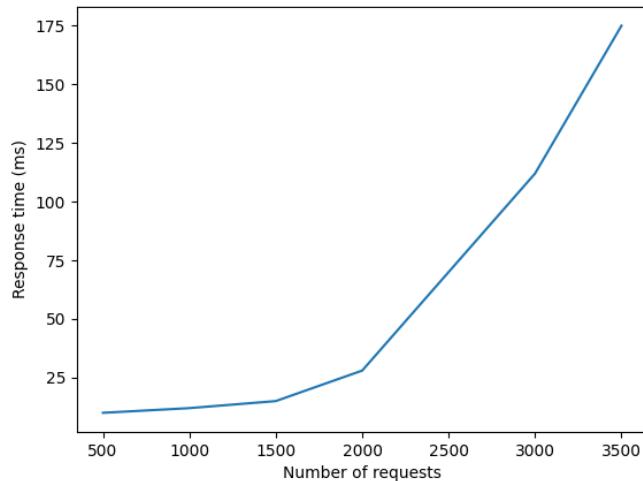
Attraverso uno script Python sono stati disegnati i grafici di Response Time, Throughput e Power, definita come rapporto tra Throughput e Response Time. In questo modo è stato possibile riconoscere i seguenti punti caratteristici del sistema:

- knee capacity: punto in cui si ha un buon compromesso tra throughput (alto) e response time (basso); in particolare, corrisponderà al massimo valore della potenza;
- usable capacity: punto in cui si raggiunge il throughput massimo del sistema, senza superare uno specifico limite di tempo di risposta; rap-

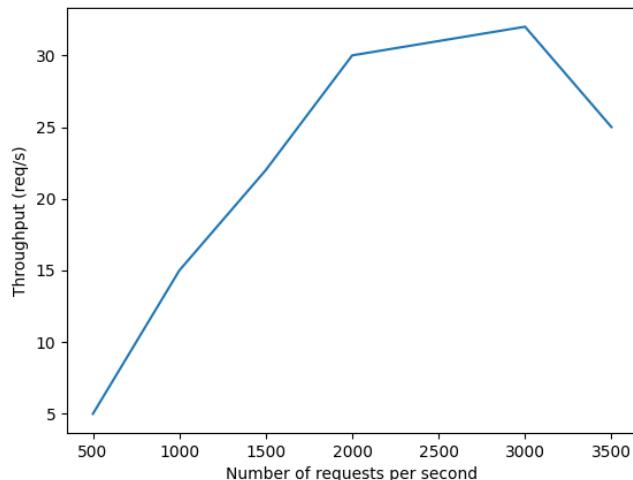
presenta il punto limite di funzionamento del sistema, oltre il quale il throughput si riduce drasticamente.

I risultati ottenuti sono i seguenti:

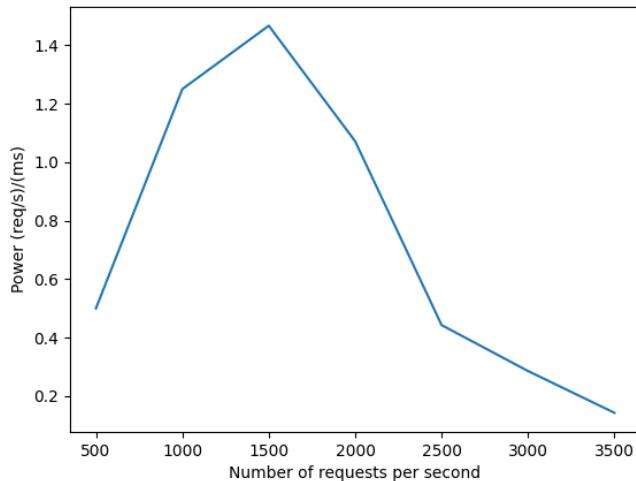
Il response time, che si mantiene inizialmente molto basso, inizia a crescere rapidamente dopo che il carico supera le 2000 richieste per minuto.



Il throughput cresce inizialmente in maniera rapida, si assesta quando il carico supera le 2000 richieste al minuto ed inizia a decrescere, a causa della saturazione del sistema, dopo le 3000 richieste al minuto



L'andamento della power è stato utilizzato per stabilire quindi i valori di knee ed usable capacity associati al sistema:



In accordo con il valore di massimo della power il valore della knee capacity per il nostro sistema è quindi pari a 1500 richieste al minuto. Il valore della usable è invece prossimo a 3000 richieste per minuto, valore limite oltre il quale si ha una rapida degenerazione delle performance. A partire dai dati raccolti si può facilmente calcolare la variazione percentuale delle metriche misurate in corrispondenza della knee e della usable capacity. Il throughput in particolare aumenta di circa il 63%, mentre il response time arriva quasi a triplicarsi.

2.1.1 Indice di equità

E' stato creato tramite Apache JMeter un test plan contenente diversi thread group, che richiedono in modo concorrente ad uno stesso Web Server le 12 risorse menzionate precedentemente.

Attraverso il CTT di questi thread group, sono stati specificati i seguenti fair throughput:

- Primo thread group: 800 richieste/min;

- Secondo thread group: 200 richieste/min;
- Terzo thread group: 500 richieste/min;

Mediante l'utilizzo di un summary report è stato invece raccolto il parametro del throughput misurato, per andare così a calcolare la quantità del throughput normalizzato.

Sono stati quindi misurati i seguenti throughput:

- Primo thread group: 195,48 richieste/min;
- Secondo thread group: 201,5826 richieste/min;
- Terzo thread group: 502,8612 richieste/min;

I throughput normalizzati x_i , dati dal rapporto tra throughput misurati e fair, sono i seguenti:

- Primo thread group: 0.244;
- Secondo thread group: 1.007;
- Terzo thread group: 1.005;

Si può quindi calcolare il fairness index:

$$FairnessIndex = \frac{(0.244 + 1.007 + 1.005)^2}{(3(0.244^2 + 1.007^2 + 1.005^2))} = 0.81$$

Si può osservare come il Web Server sia fair con gli ultimi due thread group, caratterizzati da un CTT basso, e lo sia molto meno con il primo thread group, per il quale si misura un throughput molto più basso rispetto a quello assegnato su JMeter.

2.2 Workload Characterization

L'obiettivo della workload characterization è quella di analizzare ed estrarre parametri statistici che siano caratteristici del workload. Una buona analisi

consente di descrivere dettagliatamente un workload con l'utilizzo di un insieme di parametri sintetici che allo stesso tempo caratterizzano gran parte della variabilità del dataset. La workload characterization da noi eseguita può essere riassunta nei seguenti passi:

- Determinazione del SUT;
- Generazione di un workload di analisi che sia quanto più realistico possibile;
- Collezionamento dei parametri di alto e di basso livello;
- Generazione di un workload sintetico a partire dai parametri di alto livello raccolti precedentemente;
- Validazione del workload sintetico.

In figura 2.1 si possono visualizzare le varie fasi dell'esercizio:

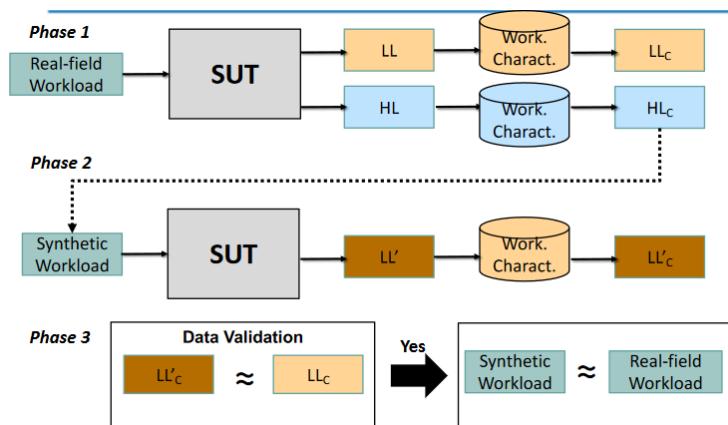


Figura 2.1: Fasi dell'homework

Innanzitutto, bisogna specificare che il SUT è lo stesso Web Server dell'esercizio precedente, con l'unica differenza nel numero di risorse memorizzate. Infatti, in questo caso sono state posizionate 50 risorse di dimensioni differenti, che spaziano dall'1KB fino ad arrivare a 20MB. Nel collezionamento dei parametri durante il testing, è necessario distinguere tra due diverse categorie di parametri:

- parametri di alto livello: tutti quei parametri raccoglibili lato client, come throughput e response time, che non forniscono una chiara indicazione su cosa sta accadendo lato server;
- parametri di basso livello: parametri recuperati direttamente dal sistema sotto test e consistono in informazioni riguardanti lo stato del servente, come l'utilizzo dei processori e della memoria.

Il workload reale, da cui verranno raccolti dati sia di alto livello (HL) che di basso livello (LL), è stato generato tramite il tool Apache JMeter.

A differenza del test plan configurato nell'esercizio precedente, qui sono stati creati ben 5 thread group, ognuno caratterizzato da 10 richieste HTTP differenti. Inoltre, il test plan è stato impostato in modo tale che i vari thread group vengano runnati consecutivamente (uno alla volta) in modo da rendere indipendente la loro esecuzione.

Di conseguenza, ogni thread group avrà una duration di 60 secondi, 20 thread e ramp-up unitario. Un'altra differenza con l'esercizio precedente è che in questo caso il valore del CTT è una costante pari a 1500. Infine, i parametri di alto livello sono raccolti direttamente dal tool grazie ad un listener di tipo Simple Data Writer.

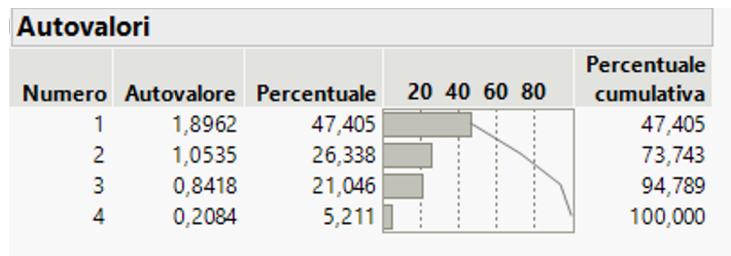
Workload Characterization HL Come prima operazione è stato deciso di applicare un pre-filtering sui dati direttamente su JMP.

id	timeStamp	elapsed	label	responseCode	responseMessage	threadName	dataType	success	failureMessage	bytes	sentBytes	grpThreads	allThreads	URL
1	1.66668e+12	26	HTTP Request 2	200	OK	Thread Group 1..1..bin	true			1335	168	1	1	http://192.168...
2	1.66668e+12	23	HTTP Request 5	200	OK	Thread Group 1..1..bin	true			5773	175	2	2	http://192.168...
3	1.66668e+12	50	HTTP Request 11	200	OK	Thread Group 1..1..text	true			2311	173	2	2	http://192.168...
4	1.66668e+12	2	HTTP Request 10	200	OK	Thread Group 1..1..text	true			2311	173	3	3	http://192.168...
5	1.66668e+12	3	HTTP Request 9	200	OK	Thread Group 1..1..text	true			7380	175	3	3	http://192.168...
6	1.66668e+12	2	HTTP Request 5	200	OK	Thread Group 1..1..bin	true			5774	175	4	4	http://192.168...
7	1.66668e+12	37	HTTP Request 3	200	OK	Thread Group 1..1..bin	true			836599	168	4	4	http://192.168...
8	1.66668e+12	14	HTTP Request 2	200	OK	Thread Group 1..1..bin	true			1334	168	5	5	http://192.168...
9	1.66668e+12	2	HTTP Request 10	200	OK	Thread Group 1..1..text	true			2310	173	5	5	http://192.168...
10	1.66668e+12	3	HTTP Request 13	200	OK	Thread Group 1..1..bin	true			836599	168	7	7	http://192.168...
11	1.66668e+12	35	HTTP Request 7	200	OK	Thread Group 1..1..text	true			11229	171	7	7	http://192.168...
12	1.66668e+12	90	HTTP Request 6	200	OK	Thread Group 1..1..bin	true			519375	171	8	8	http://192.168...
13	1.66668e+12	157	HTTP Request 6	200	OK	Thread Group 1..1..bin	true			519375	171	9	9	http://192.168...
14	1.66668e+12	58	HTTP Request 6	200	OK	Thread Group 1..1..bin	true			519375	171	9	9	http://192.168...
15	1.66668e+12	8	HTTP Request 8	200	OK	Thread Group 1..1..bin	true			7380	175	9	9	http://192.168...
16	1.66668e+12	2	HTTP Request 12	200	OK	Thread Group 1..1..text	true			1334	168	9	9	http://192.168...
17	1.66668e+12	2	HTTP Request 8	200	OK	Thread Group 1..1..bin	true			7380	175	9	9	http://192.168...
18	1.66668e+12	3	HTTP Request 8	200	OK	Thread Group 1..1..bin	true			73821	175	10	10	http://192.168...
19	1.66668e+12	5	HTTP Request 5	200	OK	Thread Group 1..1..bin	true			5774	175	11	11	http://192.168...
20	1.66668e+12	301	HTTP Request 9	200	OK	Thread Group 1..1..bin	true			217019	172	11	11	http://192.168...
21	1.66668e+12	1	HTTP Request 2	200	OK	Thread Group 1..1..bin	true			1334	168	12	12	http://192.168...
22	1.66668e+12	32	HTTP Request 1	200	OK	Thread Group 1..1..bin	true			166609	170	13	13	http://192.168...
23	1.66668e+12	1	HTTP Request 7	200	OK	Thread Group 1..1..text	true			11229	171	12	12	http://192.168...
24	1.66668e+12	2	HTTP Request 10	200	OK	Thread Group 1..1..text	true			2311	173	13	13	http://192.168...
25	1.66668e+12	1	HTTP Request 5	200	OK	Thread Group 1..1..bin	true			5773	175	13	13	http://192.168...
26	1.66668e+12	23	HTTP Request 3	200	OK	Thread Group 1..1..bin	true			836599	168	13	13	http://192.168...
27	1.66668e+12	3	HTTP Request 8	200	OK	Thread Group 1..1..bin	true			7380	175	13	13	http://192.168...
28	1.66668e+12	3	HTTP Request 8	200	OK	Thread Group 1..1..bin	true			73821	175	14	14	http://192.168...
29	1.66668e+12	5	HTTP Request 7	200	OK	Thread Group 1..1..text	true			11229	171	15	15	http://192.168...
30	1.66668e+12	41	HTTP Request 6	200	OK	Thread Group 1..1..bin	true			519374	171	16	16	http://192.168...

Figura 2.2: Raw Data HL

Come si può notare dalla figura 2.2, sono state eliminate le colonne timeStamp, responseCode, failureMessage e IdleTime perché costanti e le colonne threadName, dataType, URL e responseMessage in quanto aventi valori categorici.

Successivamente, sono stati applicati PCA e clustering ai dati ottenuti dal pre-filtering. Sono state selezionate 3 componenti principali per ottenere una varianza del 94% e 15 cluster, attraverso il metodo di Ward. In questo modo, la varianza spiegata ha raggiunto il valore di dell'81%. Nelle seguenti figure si possono analizzare i riepiloghi delle operazioni di PCA e clustering:



Clusterizzazione gerarchica

Cronologia di clusterizzazione

Numeri di cluster	Distanza	Leader	Subordinato
28	5,50705813	1521	3077
27	5,75664846	1067	1835
26	5,95597612	3021	3048
25	6,00897010	1576	4677
24	6,21722107	4535	4538
23	6,56150830	1	1579
22	6,65647643	3	11
21	6,69667404	10	16
20	7,07294193	1576	1585
19	7,78454843	2	3
18	8,80871175	7	3009
17	9,29637616	10	1521
16	9,43776102	13	1067
15	10,85025474	1576	3021
14	11,19695178	4	6
13	11,64674081	4535	4657
12	13,39272266	7	1518
11	13,60037107	2	4
10	17,97558851	3012	4633
9	18,01054850	1	7
8	20,47797401	2	66
7	20,66315757	2	14
6	27,69967641	1	10
5	29,01969320	13	3012
4	34,11264081	1	1576
3	54,96193919	1	13
2	66,54783389	1	4535
1	75,45517258	1	2

A partire da questa caratterizzazione è stato realizzato un workload sintetico contenente esclusivamente una richiesta associata ad ogni cluster. Per la scelta dell'elemento più rappresentativo di ogni cluster è stato implemen-

tato uno script python in grado di identificare, per ogni cluster, la richiesta più frequente in esso contenuta.

Workload Characterization LL I dati di basso livello sono stati raccolti mediante l'utilizzo del comando vmstat sulla macchina virtuale Linux che ospita il Web Server. Il raccoglimento dei parametri è avvenuto con un tempo di campionamento pari ad 1 secondo ed un tempo totale pari a 310 secondi, leggermente più lungo del tempo lato client per la natura asincrona dell'esecuzione manuale del test.

Anche in questo caso, è stato applicato un pre-filtering dei dati su JMP.

		r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st	
1	6	0	247456	73372	11840	415864	51	641	4344	1500	1851	1119	25	27	16	33	0		
2	0	0	247456	73120	11840	415884	0	0	0	0	176	480	13	3	84	0	0		
3	0	0	247456	73120	11840	415884	0	0	0	0	97	206	1	0	99	0	0		
4	0	0	247456	73120	11840	415884	0	0	0	0	313	1072	19	9	72	0	0		
5	0	0	247456	73120	11840	415884	0	0	0	0	248	782	11	3	86	0	0		
6	0	1	247456	67576	11848	419604	0	0	3720	32	969	504	3	16	55	26	0		
7	5	0	250024	81372	10400	408320	0	2716	6692	2716	2119	784	6	74	17	3	0		
8	0	0	1 250024	81372	10540	418252	60	0	10180	0	3967	686	2	49	19	29	0		
9	0	0	249768	69276	10552	41842	56	0	188	48	1107	580	4	22	70	4	0		
10	0	0	249768	69276	10552	41842	16	0	16	0	4041	683	2	44	53	0	0		
11	1	0	249768	69276	10556	41844	0	0	4	332	5983	731	3	47	50	0	0		
12	0	0	249768	69024	10556	418452	0	0	8	0	7466	741	2	57	40	1	0		
13	1	0	249768	69024	10556	418512	0	0	56	0	5617	646	1	52	46	0	0		
14	2	0	249768	69024	10564	418516	0	0	0	68	4192	535	1	36	44	18	0		
15	2	0	249768	69024	10564	418520	0	0	0	0	4005	549	4	37	60	0	0		
16	0	1	249768	68288	10564	419112	0	0	592	324	8776	769	4	79	18	0	0		
17	1	0	249768	68288	10564	419336	96	0	348	0	3027	680	3	42	52	3	0		
18	3	0	249768	68288	10564	419432	0	0	64	0	5871	682	4	73	14	10	0		
19	2	0	249768	68288	10572	419436	0	0	0	60	12020	929	0	87	13	0	0		
20	0	0	249768	68288	10572	419440	0	0	0	0	5080	540	0	41	59	0	0		
21	3	0	249768	68288	10572	419440	0	0	4	0	994	412	4	15	81	0	0		
22	0	0	249768	68288	10572	419448	0	0	0	0	8600	672	1	66	33	0	0		
23	2	0	249768	68288	10572	419452	0	0	0	0	3145	487	2	30	67	0	0		
24	2	0	249768	68288	10572	419456	0	0	0	16	4861	558	2	49	48	1	0		
25	2	0	249768	68288	10580	419456	0	0	0	0	5310	667	1	63	36	0	0		
26	0	0	249768	68288	10580	419460	0	0	0	0	2985	496	3	39	58	0	0		
27	1	0	249768	68288	10580	419464	0	0	0	0	1078	424	8	22	70	0	0		
28	1	0	249768	68288	10580	419468	0	0	0	0	6144	725	5	63	32	0	0		
29	1	0	249768	68288	10588	419472	0	0	0	0	28	1923	521	3	40	55	1	0	
30	1	0	249768	68288	10588	419476	0	0	0	0	9851	843	4	71	25	0	0		
31	1	0	249768	68288	10588	419476	0	0	0	0	6996	687	2	65	33	0	0		

Figura 2.3: Raw Data LL

Dalla figura 2.3 si può vedere come vi siano delle colonne con valori costanti, come swpd, si, so e st. Inoltre, analizzando la colonna id, sono stati riscontrati dei valori elevati per 10 righe che corrispondono proprio ai 10 secondi extra, durante i quali vmstat era ancora in esecuzione ma le richieste lato client erano già finite o dovevano ancora iniziare. In questo caso, queste righe sono state considerate come degli outlier e quindi verranno eliminate dalle analisi successive.

In seguito è stata quindi applicata l'analisi delle componenti principali, per cui ne sono state selezionate ben 8 per spiegare l'89% della varianza.

Workload Characterization LL1 A partire dalla caratterizzazione di alto livello, è stato generato un workload sintetico composto da solo 15 richieste, considerando che all'inizio si era partiti con 50 richieste HTTP.

Il carico è stato sottoposto al sistema sotto test sempre tramite il tool Apache JMeter. Le impostazioni del test plan nel tool rimangono le stesse di quelle descritte all'inizio, con l'unica differenza che ora in ogni thread group ci saranno meno richieste.

Lo scopo di questa seconda fase è quello di raccogliere i dati di basso livello quando al SUT viene applicato il workload sintetico prescelto. I parametri di basso livello sono sempre raccolti con il comando vmstat, in maniera analoga a quanto descritto precedentemente.

Importati i dati su JMP, si può procedere con il pre-filtering, che consiste nell'eliminare le colonne con deviazione standard molto bassa o nulla come swpd, si, so e st. Inoltre, sono state rimosse anche qui le righe dei 10 secondi in cui il carico non è imposto al sistema.

		r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st
1	1	0	250176	69136	16264	446856	93	862	5629	1773	455	1099	26	16	17	41	0	
2	0	0	250176	69136	16264	446860	0	0	0	0	108	267	1	0	99	0	0	
3	0	0	250176	69136	16264	446860	0	0	0	0	96	208	1	0	99	0	0	
4	1	2	250176	69126	16264	447040	0	0	180	0	86	161	0	0	92	8	0	
5	4	0	250176	62392	16280	450856	300	0	4136	320	1936	1594	3	23	2	71	0	
6	5	0	250176	72004	16276	442820	0	0	9360	0	10077	1236	2	94	3	0	0	
7	4	0	250176	74104	16276	439276	0	0	4720	0	10851	1197	3	91	4	1	0	
8	7	0	250176	73186	16276	439280	0	0	0	0	10226	985	4	82	14	0	0	
9	2	0	250176	69136	16276	439420	0	0	144	0	10231	1228	10	85	5	0	0	
10	12	0	250176	68700	16284	439416	0	0	0	56	10749	1099	4	85	11	0	0	
11	1	0	250176	63524	16300	439732	12	0	320	48	8011	977	5	87	8	0	0	
12	1	0	250176	63524	16300	439728	0	0	0	0	7016	793	7	79	14	0	0	
13	2	0	250176	63524	16300	439732	0	0	0	0	7093	829	4	82	14	0	0	
14	3	0	250176	63524	16300	439732	0	0	0	0	7506	790	3	83	14	0	0	
15	12	2	250176	63524	16308	441324	60	0	1652	56	6789	928	4	82	10	4	0	
16	5	1	249920	63524	16320	443148	484	0	2168	0	6670	1276	8	90	0	2	0	
17	4	0	249920	63272	16328	443696	16	0	564	0	7239	979	6	88	6	0	0	
18	9	0	249920	63272	16328	443696	0	0	0	0	7469	762	1	88	11	0	0	
19	2	0	249920	63272	16328	443700	0	0	0	0	7469	784	5	80	15	0	0	
20	5	0	249920	63272	16336	443692	0	0	0	0	7833	884	13	82	6	0	0	
21	2	0	249920	63272	16336	443704	0	0	0	0	7467	860	10	83	7	0	0	
22	5	0	249920	63272	16336	443708	0	0	0	0	7712	785	4	84	12	0	0	
23	2	0	249920	63272	16336	443708	0	0	0	0	7898	833	5	81	14	0	0	
24	8	0	249920	63272	16336	444468	0	0	642	0	6782	880	3	90	8	0	0	
25	1	0	250176	67886	16344	442428	0	388	168	404	7900	2173	8	88	4	0	0	
26	4	0	250176	62468	16348	443192	136	68	2672	68	5823	1154	2	95	1	2	0	
27	13	0	250176	61460	16504	443392	32	0	340	0	6907	1029	15	82	0	3	0	
28	11	0	250176	61460	16504	443388	0	0	0	0	7672	853	15	85	0	0	0	
29	10	0	250176	61208	16504	443392	88	0	88	0	6890	891	20	80	0	0	0	
30	4	1	250176	72516	16512	445628	0	0	2360	100	6802	819	16	80	0	3	0	
31	8	0	250432	74672	16512	439856	16	240	1489	240	6075	1358	3	97	0	0	0	

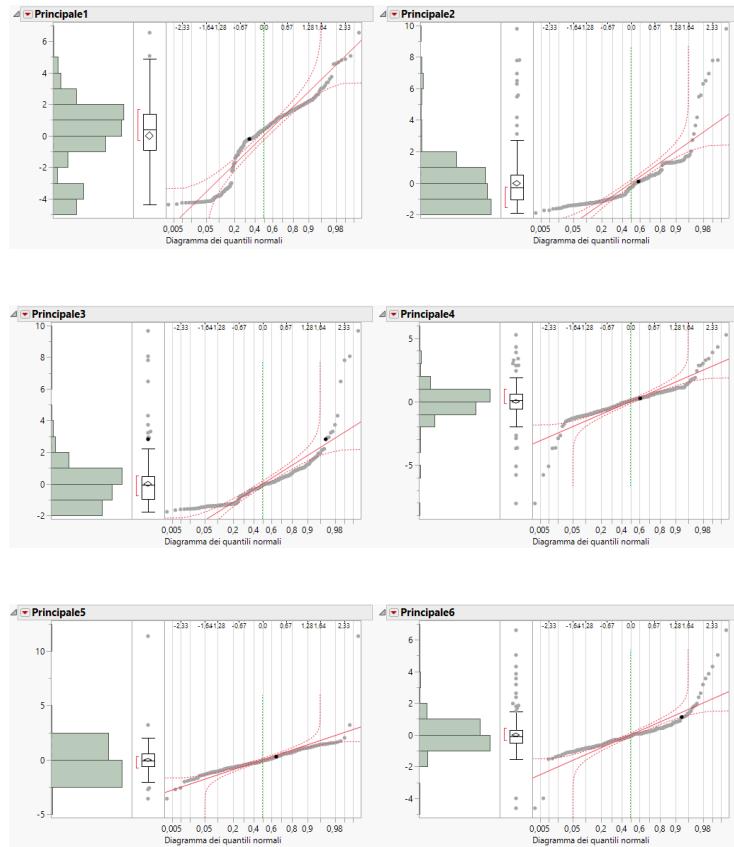
Dopo il pre-filtering è stata applicata la stessa analisi delle componenti principali che era stata fatta sul workload di basso livello LL, scegliendo quindi 8 componenti principali. Si noti che viene eseguita la stessa analisi in modo tale che si possano confrontare due dataset con lo stesso numero di colonne nella fase successiva.

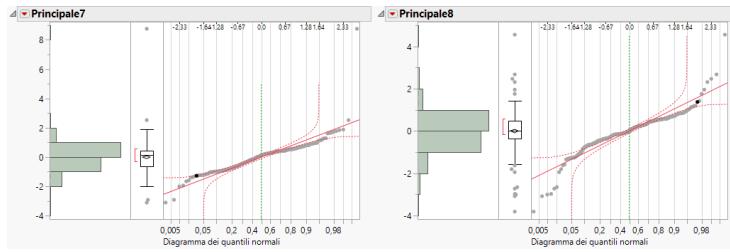
Confronto tra LL e LL1 L'ultima fase di questo homework prevede il confronto tra i due workload LL e LL1, per verificare se essi siano statisticamente differenti o meno. Come mostrato nella figura 2.1, tale analisi potrà verificare che le richieste del workload sintetico approssimino bene le richieste fatte dal workload reale iniziale.

La data validation tra i due workload consiste in due fasi:

- test visivo della normalità per ognuna delle 8 componenti principali, allo scopo di verificare se i dati sono distribuiti secondo una normale;
- applicazione di un test parametrico o non parametrico in base al risultato del test svolto nella prima fase.

In questo esercizio, il test visivo della normalità ha stabilito che tutte e 8 le componenti principali non sono distribuite normalmente. Di conseguenza, è stato applicato un test non parametrico per ognuna delle componenti attraverso uno script Python.





Sfruttando la funzione ranksums del modulo `scipy.stats`, è stato eseguito il test di Wilcoxon. Tale test, effettuato per ogni componente principale, ritorna in output un p-value alto, indicando che il test non può rigettare l'ipotesi nulla e affermando che le componenti dei due carichi non sono statisticamente differenti.

```
pi C:\Users\ltama\Desktop\Impianti di Elaborazione\Homework\Workload characterization> python .\ranksum_test.py "C:\Users\ltama\Desktop\Impianti di Elaborazione\Homework\Work Server\Workload characterization\workloadLL.txt" "C:\Users\ltama\Desktop\Impianti di Elaborazione\Homework\Work Server\Workload characterization\workloadLL.txt"
Il p-value per la componente principale 1 è pari al: 0.2989992343037669
Il p-value per la componente principale 2 è pari al: 0.698628872309221
Il p-value per la componente principale 3 è pari al: 0.4541958011988395
Il p-value per la componente principale 4 è pari al: 0.3333333333333333
Il p-value per la componente principale 5 è pari al: 0.7591258433333333
Il p-value per la componente principale 6 è pari al: 0.2571070611958892
Il p-value per la componente principale 7 è pari al: 0.8867861645014661
Il p-value per la componente principale 8 è pari al: 0.7323848794023213
```

Per tutte le componenti analizzate l'ipotesi nulla non può quindi essere rigettata per il livello di confidenza del 95% considerato, possiamo quindi concludere che le feature di basso livello estratte non sono statisticamente differenti e provengono quindi dalla stessa popolazione.

Componente	P-Value
1	0.299
2	0.6986
3	0.4541
4	0.3333
5	0.7591
6	0.2572
7	0.8867
8	0.7324

Tabella 2.1: Risultati del ranksum test

Otteniamo quindi la conferma che il workload sintetico generato a partire dalla caratterizzazione statistica del workload di alto livello produce effetti statisticamente equivalenti a quello reale sul sistema in analisi.

2.3 Design of Experiment

Il design of experiment permette di determinare quali sono i fattori che più influiscono su una particolare metrica di performance misurata sul SUT. In un esperimento si va a valutare quindi come viene modificata una variabile di output al variare dell'input. Le variabili che influenzano l'uscita prendono il nome di fattori e ad ogni fattore sono associati dei livelli, ovvero i valori che il fattore può assumere.

In questo esercizio il SUT rimane lo stesso descritto nei primi due paragrafi e le richieste HTTP sottomesse al sistema avverranno sempre tramite Apache JMeter. Nella configurazione del DoE sul tool JMP è stato innanzitutto creato un piano personalizzato. Il design è caratterizzato 2 fattori:

- Intensità: indica il carico imposto al sistema in termini di CTT su JMeter; il throughput può essere 750, 1500 o 2250 richieste per minuto;
- Tipo Pagina: dimensione delle pagine richieste al web server; le dimensioni possono essere Small (0-50KB), Medium (50KB-500KB) e Big (500KB-10MB).

Per la variabile di risposta è stata scelta il response time che ci impiega il server a servire una risorsa con un determinato request rate.

Inoltre, si osservi come per i livelli dell'intensità siano stati scelti rispettivamente il 25%, il 50% e il 75% dell'usable capacity trovata nel primo esercizio, pari al valore di 3000 req/min.

Il numero di ripetizioni scelto per tale design è pari a 5. Di conseguenza, si parlerà di un full factorial design, con 2 fattori da 3 livelli e 5 ripetizioni,

per cui, in totale, si avranno $3^2 * 5 = 45$ esperimenti da condurre. Di seguito si riporta una parte della tabella degli esperimenti con i relativi output:

		Intensità	Tipo Pagina	Elapsed
1	750	Big		3782,1901
2	750	Big		3878,8354
3	750	Big		4371,0384
4	750	Big		4817,5445
5	750	Big		6106,1345
6	750	Medium		637,0364
7	750	Medium		24,9227
8	750	Medium		9,1065
9	750	Medium		5,9217
10	750	Medium		6,4750
11	750	Small		2,5392
12	750	Small		2,4994
13	750	Small		2,7548
14	750	Small		2,5833
15	750	Small		2,8318
16	1500	Big		21045,022
17	1500	Big		21042,089
18	1500	Big		21040,256
19	1500	Big		21038,100
20	1500	Big		21043,667
21	1500	Medium		5,9575
22	1500	Medium		6,0124
23	1500	Medium		6,5468
24	1500	Medium		6,5252
25	1500	Medium		7,1472
26	1500	Small		2,7659
27	1500	Small		2,7600
28	1500	Small		2,8947
29	1500	Small		3,0412
30	1500	Small		3,1092
31	2250	Big		21042,467
32	2250	Big		21042,400
33	2250	Big		21043,400
34	2250	Big		21039,633
35	2250	Big		21038,911
36	2250	Medium		7,6863
37	2250	Medium		8,1970
38	2250	Medium		0,0717

E' stata quindi calcolata attraverso JMP l'allocazione di variazione sia dell'intero sistema che dei singoli fattori e della loro interazione. Ciò permette di ottenere l'importanza dei fattori, con il rapporto tra varianza del singolo fattore e varianza totale. Si noti che l'importanza non è un concetto

statistico ed è compito di chi fa analisi capire cosa è importante o meno.

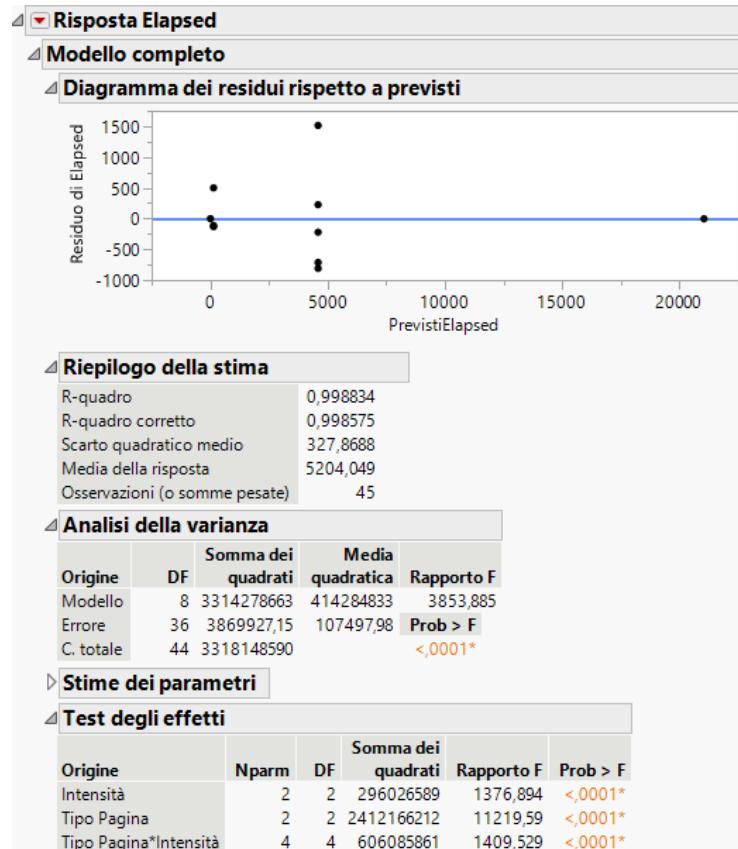


Figura 2.4: Riepilogo DoE

Come si può vedere dalla figura 2.4, sono state ottenute le somme dei quadrati relative all'intero sistema (SST), all'errore (SSE) e ai fattori e alle loro interazioni (SSA, SSB, SSAB).

E' quindi possibile andare a calcolare l'importanza, rispettivamente, per i fattori Tipo Pagina e Intensità, per la loro interazione e per l'errore:

$$\frac{SSA}{SST} = \frac{2412166212}{3318148590} \approx 0.727 = 72.7\%$$

$$\frac{SSB}{SST} = \frac{296026589}{3318148590} \approx 0.089 = 8.9\%$$

$$\frac{SSAB}{SST} = \frac{606085861}{3318148590} \approx 0.1827 = 18.27\%$$

$$\frac{SSE}{SST} = \frac{3869927.15}{3318148590} \approx 0.0011 = 0,11\%$$

Da questa analisi otteniamo una chiara descrizione dell'importanza dei fattori. Il fattore fondamentale è senza dubbio la dimensione della pagina che spiega oltre il 70% della varianza complessiva. Risulta essere inoltre abbastanza significativa anche l'interazione tra tipo di pagina ed intensità del request rate. I termini di errore risultano essere in questo caso marginali e associati ad una parte molto ridotta della varianza complessiva.

Passiamo a questo punto all'analisi della significatività statistica dei fattori. Per lo studio è stata utilizzata l'ANOVA (Analysis of Variance).

Inizialmente è stato quindi necessario, analizzando il design realizzato, determinare quale tipologia di ANOVA vada utilizzata. La scelta si basa essenzialmente su:

- la normalità della distribuzione dei residui;
- l'omoschedasticità, ovvero la costanza della varianza, della distribuzione dei residui.

Per testare la normalità dei residui è stato applicato un test visivo sul QQ plot automaticamente generato da JMP. Come si può vedere dalla figura 2.5, è facilmente verificabile che la distribuzione dei residui non è normale poiché i punti fuoriescono dalle bande di confidenza.

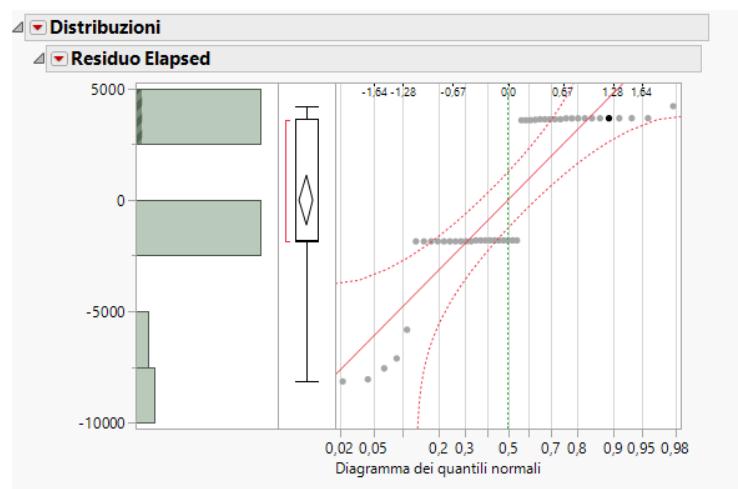


Figura 2.5: Distribuzione dei residui

Per quanto riguarda l'omoschedasticità, non è necessario effettuarne la verifica in quanto, visto che i residui non sono normali, sicuramente si dovrà optare per un test di Kruskal-Wallis, a prescindere quindi dalla presenza di questa proprietà nei residui ottenuti.

Dalla figura 2.6, che descrive il risultato del test di Kruskal-Wallis per il fattore Intensità, si può notare come si abbiano valori alti di ChiQu, per cui il fattore non è significativo. Discorso inverso si applica per il fattore Tipo Pagina, che risulta essere significativo (figura 2.7).

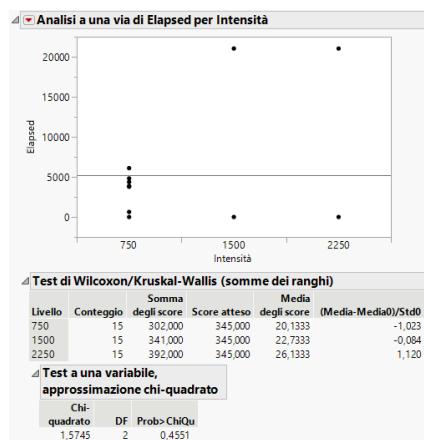


Figura 2.6: Test di Kruskal-Wallis per l'Intensità

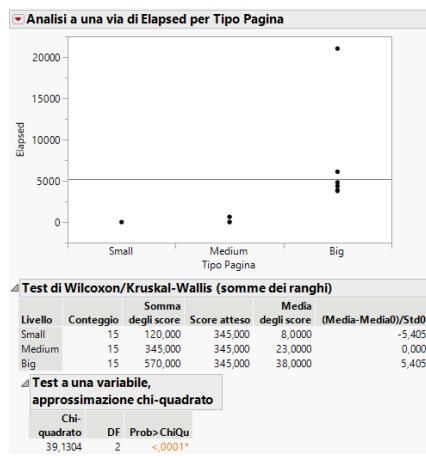


Figura 2.7: Test di Kruskal-Wallis per il Tipo Pagina

Otteniamo che quindi, per la nostra analisi, la dimensione della pagina risulti essere un fattore significativo ed importante, mentre l'intensità delle richieste è non significativo e poco importante.

Capitolo 3

Workload Characterization

Lo scopo di questo esercizio è di effettuare workload characterization a partire da dati raccolti dal Proc file system di una macchina virtuale Linux. Il Proc file system è un file system virtuale gestito dal kernel di Linux a scopo di monitoraggio delle risorse del sistema. Al suo interno si possono trovare alcuni file che descrivono lo stato corrente della macchina Linux ed altri associati allo stato dei processi in esecuzione.

```
:$ ls -l /proc
 9 root          root          0 nov 1 23
 9 root          root          0 nov 1 23
 9 root          root          0 nov 1 23
 9 so           so           0 nov 1 23
 9 so           so           0 nov 1 23
 9 root          root          0 nov 1 23
 9 so           so           0 nov 1 23
 9 so           so           0 nov 1 23
 9 so           so           0 nov 1 23
 9 root          root          0 nov 1 23
 9 root          root          0 nov 1 23
 9 so           so           0 nov 1 23
 9 so           so           0 nov 1 23
 9 so           so           0 nov 1 23
 9 rtkit         rtkit         0 nov 1 23
 9 so           so           0 nov 1 23
 9 root          root          0 nov 1 23
 9 so           so           0 nov 1 23
 9 root          root          0 nov 1 23
 9 so           so           0 nov 1 23
 9 root          root          0 nov 1 23
 9 root          root          0 nov 1 23
 9 root          root          0 nov 1 23
```

Per quest'analisi, sono stati studiati i seguenti file:

- /pid/status;
- /meminfo.

/pid/status contiene informazioni sulle risorse utilizzate da un processo, specificato dal pid nel path. /meminfo, invece, fornisce informazioni sullo stato della memoria.

```
so@so-vbox:/proc$ cat /proc/meminfo
MemTotal:          2023240 kB
MemFree:           122140 kB
MemAvailable:     1190252 kB
Buffers:           60056 kB
Cached:            1106140 kB
SwapCached:        0 kB
Active:             516640 kB
Inactive:          1181896 kB
Active(anon):      1396 kB
Inactive(anon):    527708 kB
Active(file):      515244 kB
Inactive(file):    654188 kB
Unevictable:        0 kB
Mlocked:            0 kB
SwapTotal:          1918356 kB
SwapFree:           1918356 kB
Dirty:              856 kB
Writeback:           0 kB
AnonPages:          532424 kB
Mapped:             287140 kB
Shmem:              2012 kB
KReclaimable:       60204 kB
Slab:                115708 kB
SReclaimable:       60204 kB
SUnreclaim:          55504 kB
KernelStack:         7968 kB
PageTables:          13600 kB
```

Per il campionamento di tali dati è stato scritto uno script bash che periodicamente legge i parametri da questi file del proc file system.

Nel dettaglio, si analizzano i campi di proc/pid/status:

- VmPeak: dimensione di picco della memoria virtuale
- VmSize: dimensione totale del programma

- VmHWM: limite massimo di memoria utilizzato
- VmRSS: utilizzo della memoria corrente
- VmPTE: dimensione delle entry della tabella delle pagine
- VmSwap: quantità di swap utilizzata da dati privati anonimi
- Threads: numero di thread

I campi di proc/meminfo sono molti di più e nel seguito ne viene rappresentata solo una parte:

- MemFree: quantità di RAM fisica, in kilobyte, lasciata inutilizzata dal sistema
- Buffers: quantità di RAM fisica in KB utilizzata per i buffer di file
- Cached: quantità di RAM fisica, in kilobyte, utilizzata come memoria cache
- SwapCached: quantità di swap, in kilobyte, utilizzata come memoria cache
- Active: quantità totale di memoria buffer o cache di pagina, in kilobyte, che è in uso attivo. Questa è la memoria che è stata utilizzata di recente e di solito non viene recuperata per altri scopi
- Inactive: quantità totale di memoria buffer o cache di pagina, in kilobyte, che è libera e disponibile. Questa è la memoria che non è stata utilizzata di recente e può essere recuperata per altri scopi
- Dirty: quantità totale di memoria, in kilobyte, in attesa di essere riscritta sul disco
- Writeback: quantità totale di memoria, in kilobyte, che viene riscritta attivamente sul disco

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

Si noti come lo script bash sia stato utilizzato su un processo di Firefox che eseguiva dello streaming video. Una volta ottenuti tutti i dati, si è passati al tool JMP per lavorare con la workload characterization del dataset.

Il workload da analizzare è quindi composto da 5000 campioni, per ognuno dei quali sono presenti 26 parametri (colonne). Si procederà all'analisi dei dati utilizzando prima tecniche di filtering per poi successivamente passare ad ulteriori tipologie di tecniche come PCA (Principal Component Analysis) e clustering.

	VmPeak	VmSize	VmHWM	VmRSS	VmPTE	VmSwap	Threads	MemFree	MemAvailable	Buffers	Cached	SwapCached	Active	Inactive	Unevitable	SwapFree	Dirty	Writeback	AnonP
1	2394008	2394008	92048	91940	472	0	15	260496	731896	21368	560992	9556	420922	1111052	0	1631800	10296	0	
2	2394008	2394008	92048	91940	472	0	15	253188	730016	21696	565168	9556	425512	1115364	0	1631800	11472	0	
3	2394008	2394008	92048	91940	472	0	15	244620	725120	21708	569708	9596	427932	1125308	0	1631800	11880	0	
4	2394008	2394008	92048	91940	472	0	15	235060	718982	22456	576196	9664	429422	1140312	0	1634056	12128	0	
5	2394008	2394008	92048	91940	472	0	15	210616	702632	22604	585668	9672	430672	1163008	0	1634056	10820	0	
6	2394008	2394008	92048	91940	472	0	15	193004	690756	23804	591020	9672	432544	1179364	0	1634056	10904	0	
7	2394008	2394008	92048	91940	472	0	15	178540	677240	24196	59644	10044	433082	1164284	0	1635338	10930	0	
8	2394008	2394008	92048	91940	472	0	15	178136	677092	24204	592004	10340	433164	1164532	0	1636104	10540	0	
9	2394008	2394008	92048	91940	472	0	15	177412	676388	24204	587616	10340	433260	1181564	0	1636104	10556	0	
10	2394008	2394008	92048	91940	472	0	15	177412	676388	24204	587636	10340	433260	1179308	0	1636104	10440	0	
11	2394008	2394008	92048	91940	472	0	15	178692	677676	24212	587636	10340	433260	1165552	0	1636104	10468	0	
12	2394008	2394008	92048	91940	472	0	15	192428	691456	24212	587684	10340	433316	1153292	0	1636104	10468	0	
13	2394008	2394008	92048	91940	472	0	15	211372	710316	24220	587520	10340	435400	1137936	0	1636104	10476	0	
14	2394008	2394008	92048	91940	472	0	15	211148	710096	24220	587524	10340	435400	1140336	0	1636104	10480	0	
15	2394008	2394008	92048	91940	472	0	15	210476	709440	24244	587516	10340	435400	1142756	0	1636104	9776	0	
16	2394008	2394008	92048	91940	472	0	15	187880	686884	24244	602872	10340	435480	1165252	0	1636104	9804	0	
17	2394008	2394008	92048	91940	472	0	15	181412	680424	24244	60888	10340	435480	1151916	0	1636104	9804	0	
18	2394008	2394008	92048	91940	472	0	15	180712	679748	24244	598068	10340	435512	1146052	0	1636104	9828	0	
19	2394008	2394008	92048	91940	472	0	15	180488	679524	24244	596668	10340	435512	1146996	0	1636104	9828	0	
20	2394008	2394008	92048	91940	472	0	15	179988	679040	24280	598320	10340	435512	1151360	0	1636104	9328	0	
21	2394008	2394008	92048	91940	472	0	15	179736	678840	24280	607612	10340	435560	1164008	0	1636104	9352	0	
22	2394008	2394008	92048	91940	472	0	15	179736	678840	24280	599492	10340	435556	1149992	0	1636104	9356	0	
23	2394008	2394008	92048	91940	472	0	15	179260	678368	24280	598000	10340	435560	1150540	0	1636104	9364	0	
24	2394008	2394008	92048	91940	472	0	15	178812	677924	24280	599416	10340	435556	1154348	0	1636104	9368	0	
25	2394008	2394008	92048	91940	472	0	15	178336	677472	24292	598104	10340	435560	1151512	0	1636104	7248	0	
26	2394008	2394008	92048	91940	472	0	15	178336	677472	24292	609004	10340	435560	1165968	0	1636104	7272	0	
27	2394008	2394008	92048	91940	472	0	15	178112	677248	24292	609004	10340	435572	1164476	0	1636104	7280	0	
28	2394008	2394008	92048	91940	472	0	15	177664	676816	24292	602220	10340	435580	1159648	0	1636104	7284	0	
29	2394008	2394008	92048	91940	472	0	15	177188	676328	24292	596720	10340	435564	1156872	0	1636104	7284	0	
30	2394008	2394008	92048	91940	472	0	15	176936	676100	24336	596784	10340	435572	1156756	0	1636104	5616	0	

Il pre-filtering consiste inizialmente nell'eliminare le colonne di VmSwap e Unevitable perché costanti. Successivamente, sono stati analizzati gli outlier e si è deciso di eliminare le prime 31 righe del dataset in quanto rappresentavano una fase di startup per il sistema. Infine, le colonne di VmPeak, VmSize, VmHWM e VmRSS sono state eliminate dato che avevano una deviazione standard molto bassa.

A valle di tali operazioni, si ottiene un dataset filtrato sul quale sarà possibile effettuare la PCA.

3.1 Principal Component Analysis

L'idea di base è quella di individuare nuove componenti incorrelate tra loro attraverso un cambiamento dello spazio di stato che permetta, inoltre, di ordinare le nuove componenti in base a quanta varianza esse descrivono. La PCA risulta quindi fondamentale per la nostra analisi, consente infatti in primis di decorrelare i dati raccolti, ma anche di ridurre la dimensionalità del dataset trascurando alcune delle componenti principali che spiegano una percentuale bassa della variabilità dei dati.

La PCA è stata effettuata tramite JMP, che garantisce la normalizzazione, tramite il calcolo degli z-values, prima di effettuare la trasformazione. Delle 20 componenti principali, sono state mantenute solo le prime 6. Nonostante una riduzione così drastica della dimensionalità del database viene comunque conservato oltre il 91% della varianza del campione, come si può osservare dalla figura 3.1.

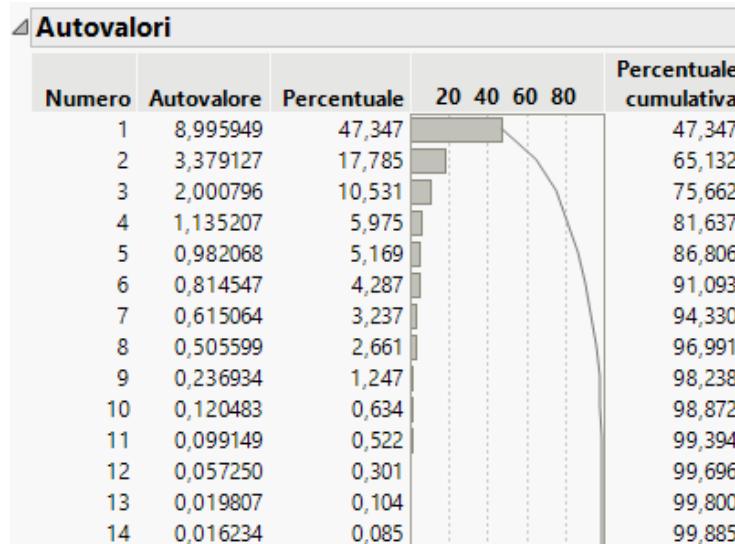
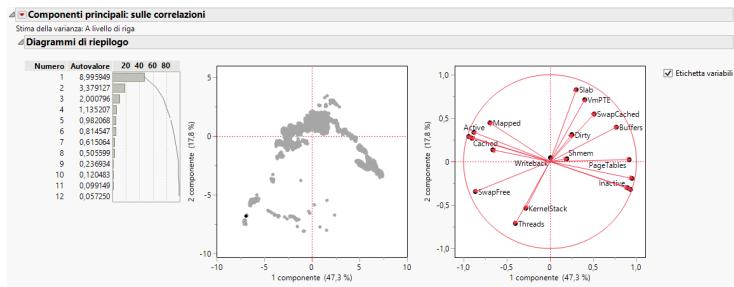


Figura 3.1: Tabella degli autovalori

Inoltre, osservando il Loading Plot generato automaticamente dal tool, si noti come la PCA abbia evidenziato una forte correlazione tra alcuni parametri, come Active, Mapped, Cached e Writeback, il che spiega l'efficacia della

trasformazione nel ridurre il numero di componenti del dataset.



3.2 Clustering

La fase successiva della caratterizzazione è il clustering, tecnica che ha lo scopo di ridurre il numero di punti presenti all'interno del dataset, raggruppando i punti tra di loro simili. Per la clusterizzazione è stato adoperato il metodo di Ward, che realizza la clusterizzazione minimizzando la devianza intra-cluster.

In questo esercizio, sfruttando la cronologia di clusterizzazione, è stato scelto un numero di cluster pari a 12, oltre il quale la distanza inter-cluster non varia in maniera significativa (figura 3.2).

Clusterizzazione gerarchica				
Cronologia di clusterizzazione				
Numeri di cluster	Distanza	Leader	Subordini	
32	6,78463504	4064	4168	
31	7,59822114	2353	3202	
30	7,63785123	359	2445	
29	8,44336256	4470	4478	
28	8,77157445	1630	1648	
27	9,0474630	244	1561	
26	9,2558540	1	30	
25	9,84462391	1630	1665	
24	9,92180846	189	223	
23	9,95348115	244	295	
22	10,12578076	286	291	
21	10,49740795	2248	2353	
20	11,28365922	2016	3196	
19	11,37807956	4470	4543	
18	11,40431546	450	2206	
17	12,28195856	1653	2248	
16	12,73777561	189	231	
15	13,3403351	4064	4339	
14	13,56432478	286	363	
13	14,20257994	359	2519	
12	21,31818242	475	2242	
11	23,24085149	1	189	
10	23,84332202	244	286	
9	25,16597040	1630	4064	
8	26,88366022	1630	2016	
7	32,01473019	359	1630	
6	40,97955916	359	1653	
5	61,06913087	244	359	
4	61,83392130	244	450	
3	63,35836042	244	4470	
2	65,70147118	1	475	
1	67,64895435	1	244	

Figura 3.2: Cronologia di clusterizzazione

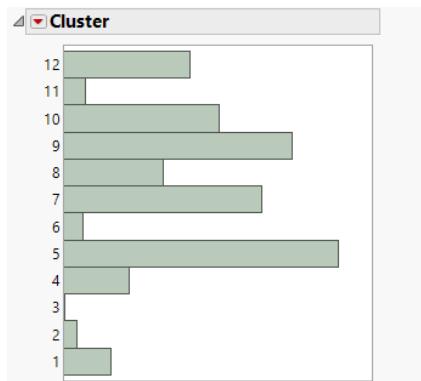
Per questa scelta del numero di cluster si ottiene una perdita di devianza del 10%. Questo valore è stato calcolato con l'ausilio di uno script MATLAB che consente di calcolare la devianza persa con l'operazione di clustering.

 **DEV_LOST_per** 0.1850

Si ottiene quindi che, riducendo drasticamente le dimensioni del workload, è stato perso, tra operazioni di PCA e clustering, circa il 18% della varianza totale. Il valore della varianza persa è infatti pari a:

$$LostPCA + Clustering = 0.09 + 0.10 * 0.91 = 0.09 + 0.091 = 0.18 = 18\%$$

Effettuato il clustering, si è analizzata la distribuzione dei punti del dataset all'interno dei vari cluster:



Frequenze		
Livello	Conteggio	Prob
1	188	0,03783
2	55	0,01107
3	7	0,00141
4	259	0,05212
5	1092	0,21976
6	76	0,01529
7	788	0,15858
8	394	0,07929
9	905	0,18213
10	618	0,12437
11	87	0,01751
12	500	0,10062
Totale	4969	1,00000

Il risultato mostra la presenza di un cluster, il numero 3, con solo 7 elementi. Questo cluster è associato a degli outlier di cui è stata analizzata l'origine. Le righe raggruppate in questo cluster sono caratterizzate da un valore molto elevato della statistica di writeback: sono quindi condizioni di carico intenso per il sistema. Abbiamo quindi optato per mantenere questo cluster nonostante i dati contenuti in esso siano degli outlier per il sistema poichè associati a condizioni di funzionamento significative per il sistema.

In conclusione, si va a realizzare un workload sintetico: da 26 colonne e 5000 righe si è passati ad un dataset con 20 colonne e solamente 12 righe. Il workload sintetico, mostrato in figura 3.3, è stato ottenuto selezionando come punto rappresentativo di ogni cluster l'elemento più vicino al centroide. Per calcolare questi elementi è stato realizzato un apposito script Python.

	VmPTE	Threads	MemFree	MemAvail able	Buffers	Cached	SwapCach ed	Active	Inactive	SwapFree	Dirty	Writeback	AnonPages	Mapped	Shmem	Slab
1	472	15	169580	670044	24752	595184	10344	456760	1168428	1636104	868	0	981132	303592	23496	132208
2	472	15	76132	546776	24324	562424	11080	435012	1274468	1599608	1580	0	1117912	298104	20200	133064
3	476	14	116760	58106	33624	540412	15676	417124	1227576	1557944	9980	8204	1064112	291508	14336	134660
4	476	14	193280	655512	28776	546420	10948	420424	1161980	1581616	2872	0	1002800	293920	17044	133856
5	476	14	193040	652820	27312	550620	10952	418272	1164400	1581616	2928	0	1000140	298928	22304	133668
6	476	14	132520	566872	34712	531112	15676	405764	1238280	1557944	4848	0	1071684	313220	35784	134064
7	476	14	121160	598272	34496	561084	14732	428340	1214728	1587504	3692	0	1041864	303604	23756	134056
8	476	14	75208	459844	43636	460636	13900	361016	1329996	1505728	3660	0	1181404	295932	24240	134900
9	476	14	167408	607712	38940	521300	15068	409966	1197072	1554608	3748	0	1040660	301464	24404	134540
10	476	14	162336	590952	37056	513980	15068	407436	1192460	1554608	4032	0	1043084	295688	18604	134264
11	476	14	79600	550084	38386	555448	14832	430752	1290364	1568744	21392	0	1126328	302880	23396	134540
12	476	14	130560	393588	38396	345204	13588	301540	1343000	1538128	4408	0	1256012	270668	24300	133180

Figura 3.3: Workload sintetico

Capitolo 4

Regressione

Un modello regressivo permette di studiare una variabile come una funzione di altre variabili:

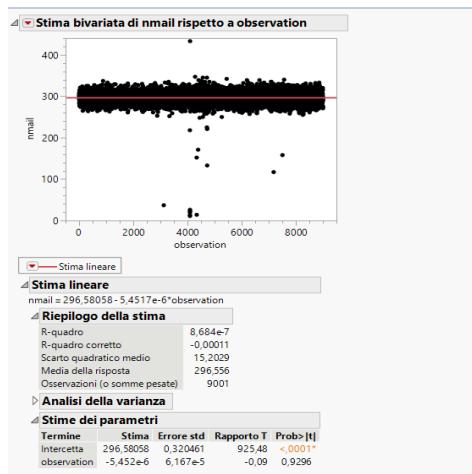
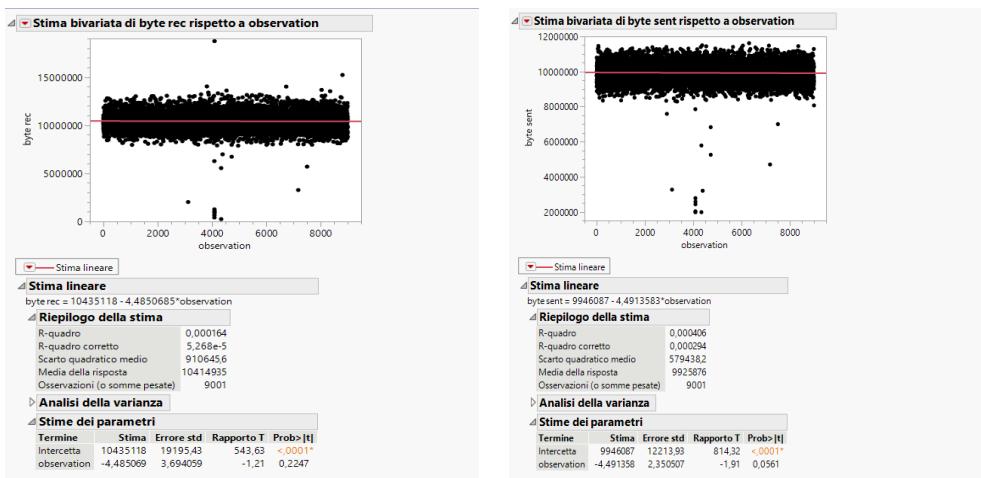
- variabile di risposta: variabile stimata
- variabili di predizione: variabili usate per predire la risposta

L'obbiettivo di questo homework è rilevare e stimare eventuali trend nelle variabili stimate, utilizzando eventualmente modelli regressivi lineari semplici, parametrici e non parametrici.

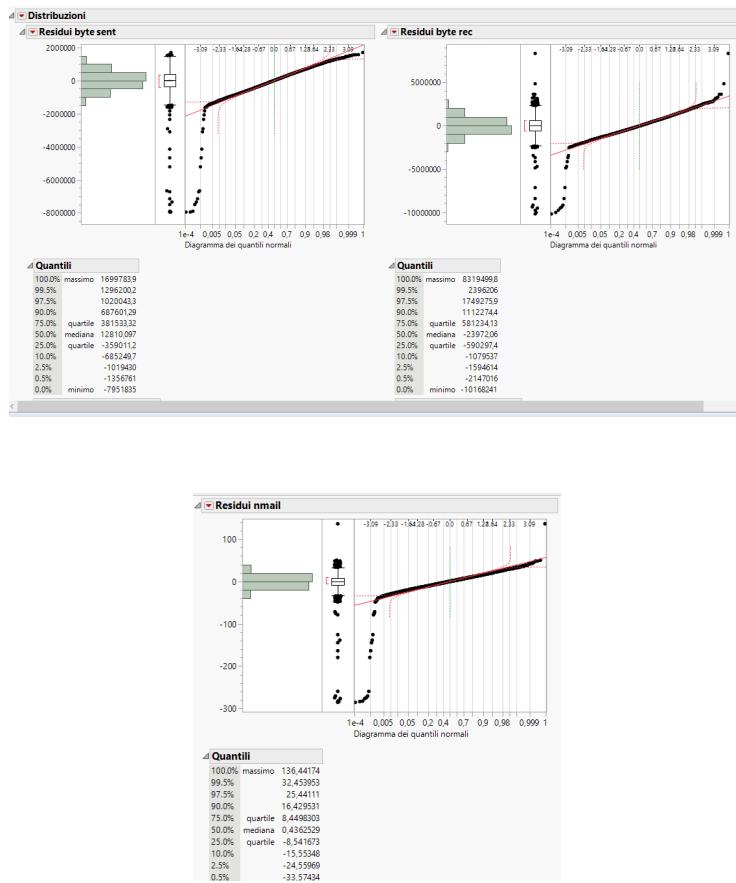
4.1 Esercizio 1 - Mail server

Innanzitutto, è stato applicato un modello regressivo lineare, per cui la risposta è una funzione lineare del predittore. Infatti, come si può notare dalle figure, sono state eseguite le stime lineari delle variabili byte rec, byte sent e nmail rispetto alla variabile di predizione observation.

Nel riepilogo della stima viene indicato l'R-quadro, ovvero il coefficiente di determinazione, che misura il legame tra la variabilità dei dati e la correttezza del modello statistico utilizzato. In tutti e 3 i casi, l'R-quadro assume un valore molto basso a dimostrazione del fatto che il modello lineare non è abbastanza corretto per predire i dati.



Successivamente, sono state analizzate le distribuzioni dei residui di byte rec, byte sent e nmail. In tutti e 3 i casi, tramite un test visivo del QQ-plot, è stato verificato che le distribuzioni fossero non normali. Di conseguenza, è stato applicato un modello non parametrico.

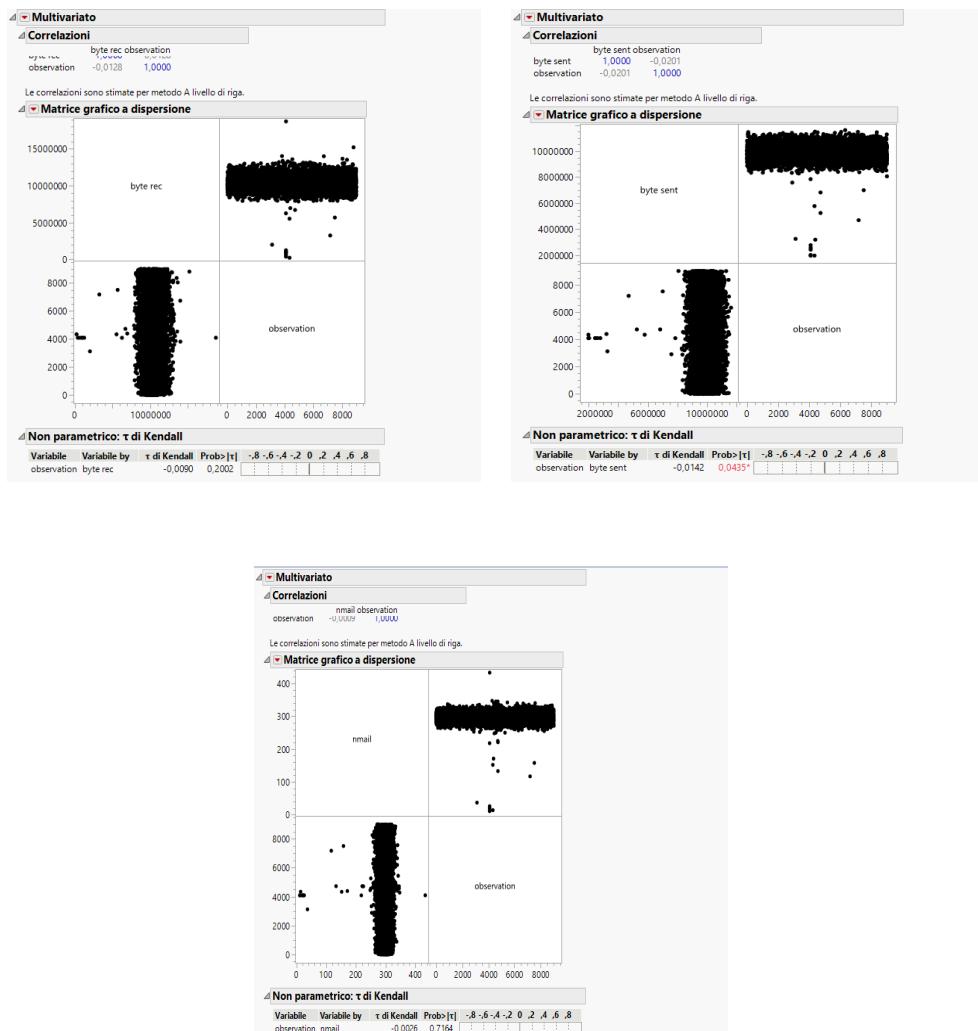


Come modello regressivo non parametrico è stato scelto il test di Mann-Kendall, che non richiede che i dati siano normalmente distribuiti o lineari. In sintesi, l'analisi darà come risultato un coefficiente di correlazione τ e un p-value. I valori di τ vanno da -1 a 1. Un valore negativo indica che le variabili sono inversamente correlate, ovvero che quando una variabile aumenta, l'altra diminuisce. I valori positivi indicano invece che quando una variabile aumenta, aumenta anche l'altra.

Un p-value inferiore o uguale a 0,05 significa che il risultato ottenuto è statisticamente significativo per l'analisi, e che l'ipotesi nulla H_0 , che afferma che non è presente un trend monotono nella distribuzione, è rifiutata.

Variabile	τ	P	H
Bytes Rec	-0.0090	0.20	0
Bytes Sent	-0.0142	0.0435	1
N Mails	-0.0026	0.7164	0

Dai dati raccolti risulta che, considerato il tasso di significatività standard pari a $\alpha = 0.05$, l'unica variabile che presenta un trend è bytes sent, mentre le altre due variabili non presentano trend significativi.



Utilizzando la procedura di Theil-Sen, implementata sfruttando uno script python, abbiamo calcolato la pendenza e l'intercetta della retta di regressione,

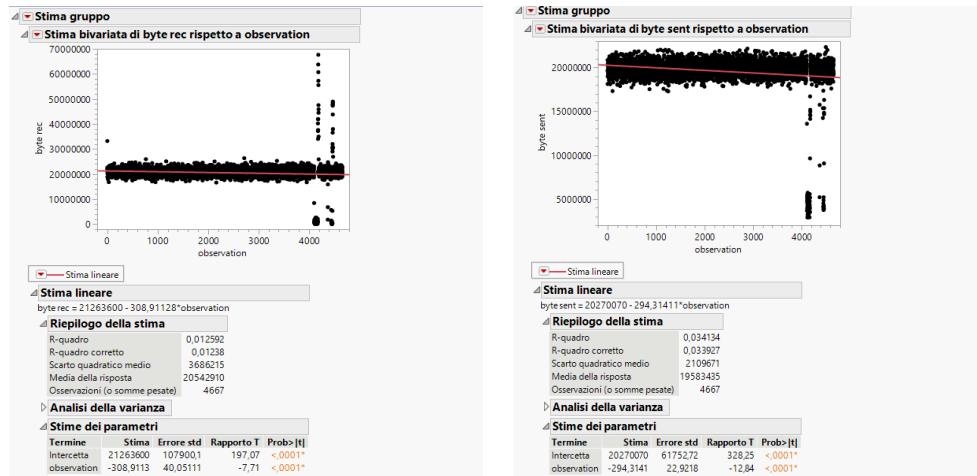
nonchè l'intervallo di confidenza associato a questo parametri:

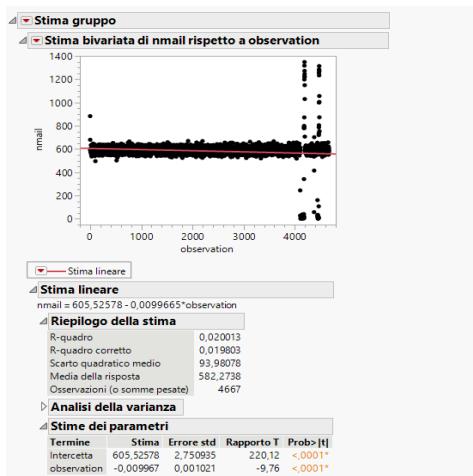
```
D:\Users\itama\Desktop\Impianti di Elaborazione\Homework\Regressione> python .\sensprocedure.py .\Homework_Regression_2022.xls @ observation byte
Slope: -4.514783992985356. Interval: [-8.891058583657588,-0.11867767792881614] Intercept: 9959247.527968435
```

4.2 Esercizio 2 - Mail Server

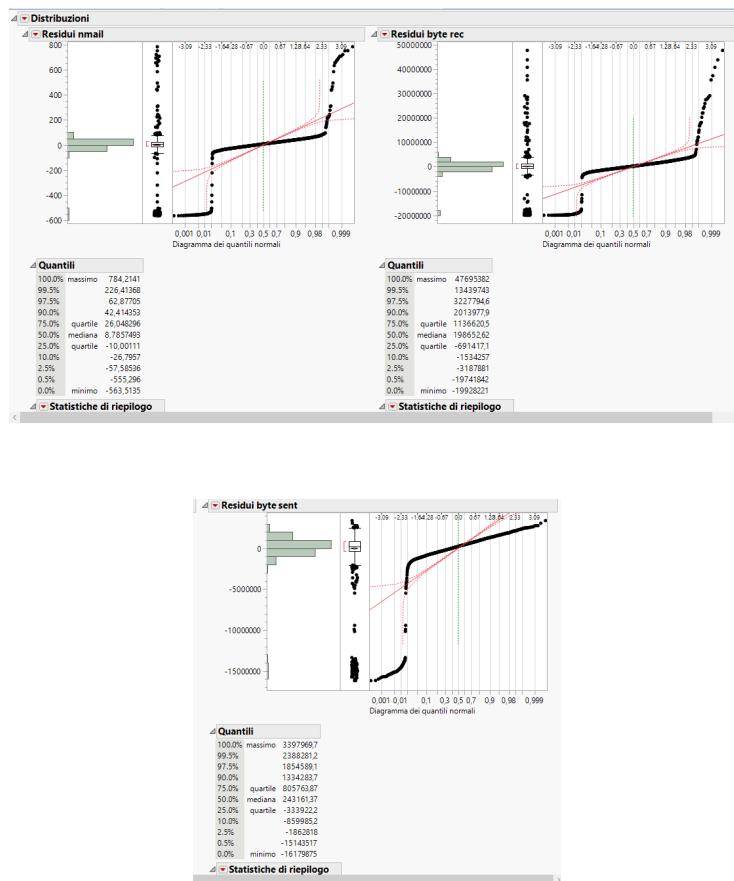
E' stato applicato un modello regressivo lineare, per cui la risposta è una funzione lineare del predittore. Infatti, come si può notare dalle figure, sono state eseguite le stime lineari delle variabili byte rec, byte sent e nmail rispetto alla variabile di predizione observation.

Nel riepilogo della stima viene indicato l'R-quadro, ovvero il coefficiente di determinazione, che misura il legame tra la variabilità dei dati e la correttezza del modello statistico utilizzato. In tutti e 3 i casi, l'R-quadro assume un valore molto basso a dimostrazione del fatto che il modello lineare non è abbastanza corretto per predire i dati.





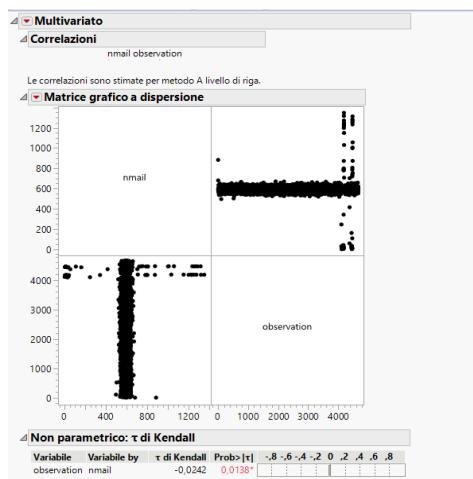
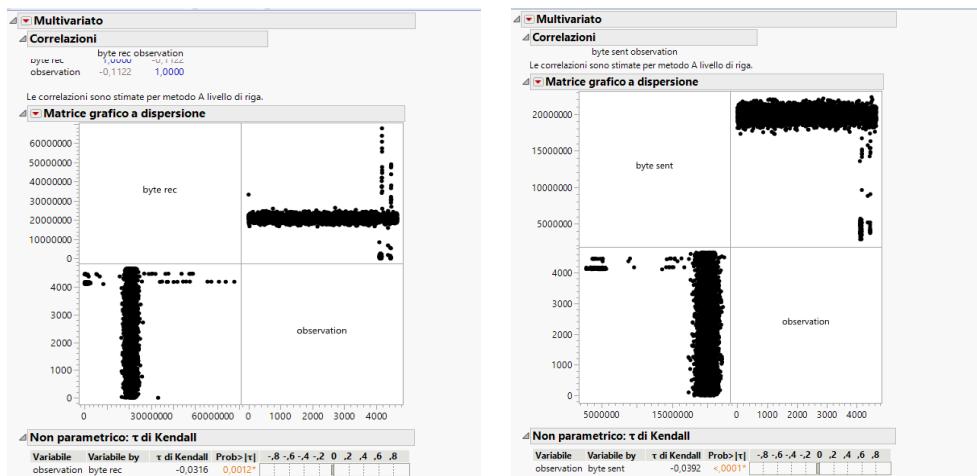
Successivamente, sono state analizzate le distribuzioni dei residui di byte rec, byte sent e nmail. In tutti e 3 i casi, tramite un test visivo del QQ-plot, è stato verificato che le distribuzioni fossero non normali. Di conseguenza, è stato applicato un modello non parametrico.



Come modello regressivo non parametrico è stato scelto il test di Mann-Kendall, che non richiede che i dati siano normalmente distribuiti o lineari.

In questo caso, tutti e 3 i test hanno dato come risultato una τ molto vicina allo 0 e p-value minori di 0.05. Il test suggerisce quindi che sono presenti dei trend, anche se con una pendenza molto ridotta.

Variabile	τ	P	H
Bytes Rec	-0.0316	0.0012	1
Bytes Sent	-0.0392	0.001	1
N Mails	-0.0242	0.0138	1



Utilizziamo anche in questo caso la procedura di Theil-Sen per il calcolo della pendenza della retta di regressione per i tre parametri:

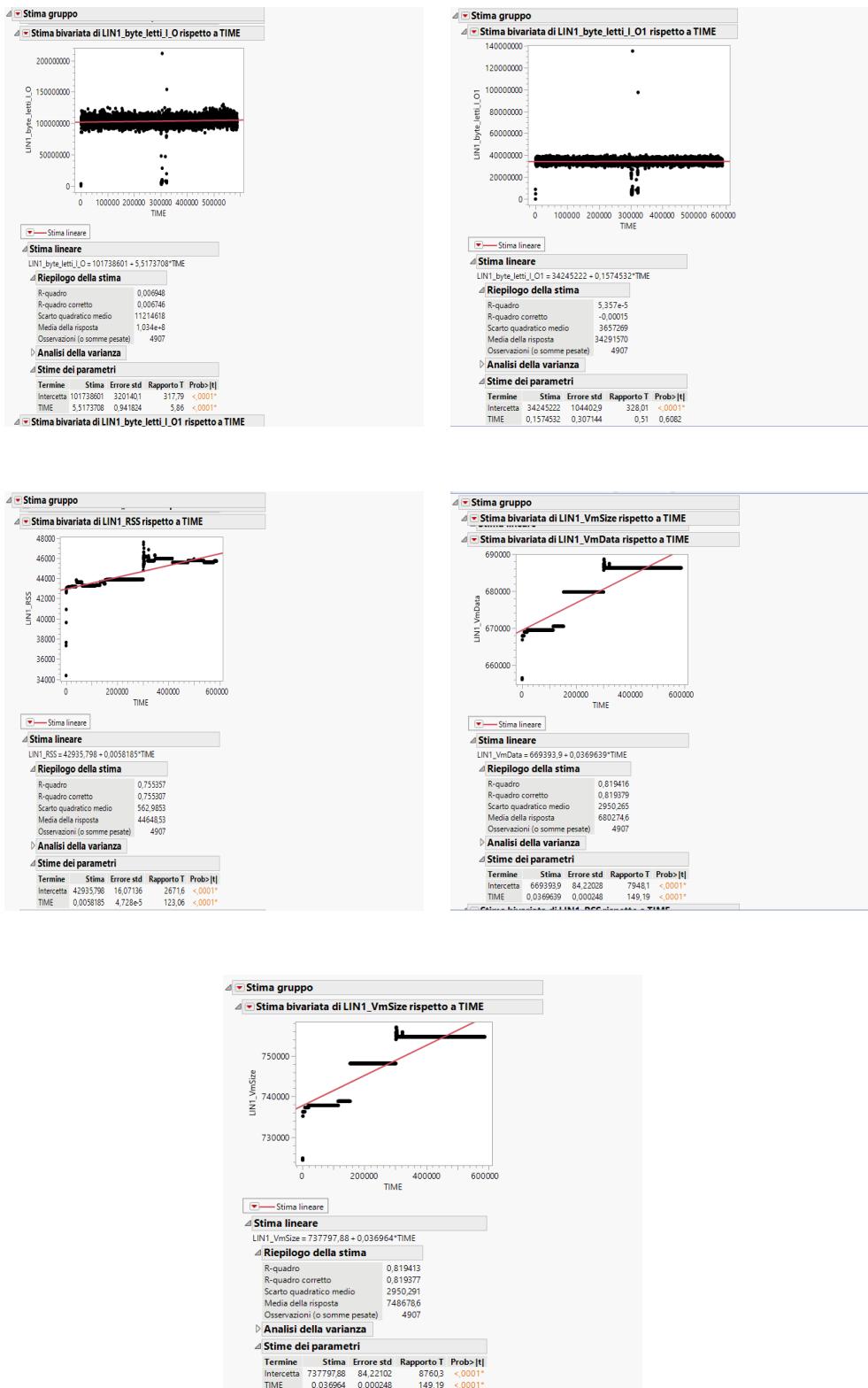
```
PS C:\Users\litalma\Desktop\Impianti di Elaborazione\Homework\Regressione> python .\sensprocedure.py .\Homework_Regression_2022.xls 1 observation nmai
1
Slope: -0.0005973715651135086. Interval: [-0.00112485039325759281, 0.0] Intercept: 592.3936678610498
PS C:\Users\litalma\Desktop\Impianti di Elaborazione\Homework\Regressione> python .\sensprocedure.py .\Homework_Regression_2022.xls 1 observation byte
1
Slope: -34.86305288658625. Interval: [-59.672672672672675, -17.370887591248876] Intercept: 19983060.10851782
PS C:\Users\litalma\Desktop\Impianti di Elaborazione\Homework\Regressione> python .\sensprocedure.py .\Homework_Regression_2022.xls 1 observation byte
_rec
Slope: -48.29366907184882. Interval: [-77.547732695689737, -18.993146773272414] Intercept: 20890131.991364423
```

4.3 Esercizio 3 - VmStat

Questo esercizio consiste nel rilevare e stimare trend su 5 variabili utilizzando modelli regressivi lineari semplici, parametrici e non parametrici. I dati sono distribuiti su 3 dataset lo scopo è confrontare i trend individuati in ognuno di essi.

Primo dataset E' stato applicato un modello regressivo lineare, per cui la risposta è una funzione lineare del predittore. Infatti, come si può notare dalle figure, sono state eseguite le stime lineari delle variabili VmSize, VmData, RSS, Byte letti IO e Byte letti IO1 rispetto alla variabile di predizione time.

Nel riepilogo della stima viene indicato l'R-quadro, ovvero il coefficiente di determinazione, che misura il legame tra la variabilità dei dati e la correttezza del modello statistico utilizzato. Nei 5 casi considerati, l'R-quadro assume un valore molto basso per le variabili Byte letti IO e Byte letti IO1 e alto per le altre 3 variabili, che quindi vengono stimate abbastanza bene con un modello regressivo lineare.



Successivamente, sono state analizzate le distribuzioni dei residui di VmSize, VmData, RSS, Byte letti e Byte scritti. In tutti e 5 i casi, tramite un test

visivo del QQ-plot, è stato verificato che le distribuzioni fossero non normali. Di conseguenza, è stato applicato un modello non parametrico.

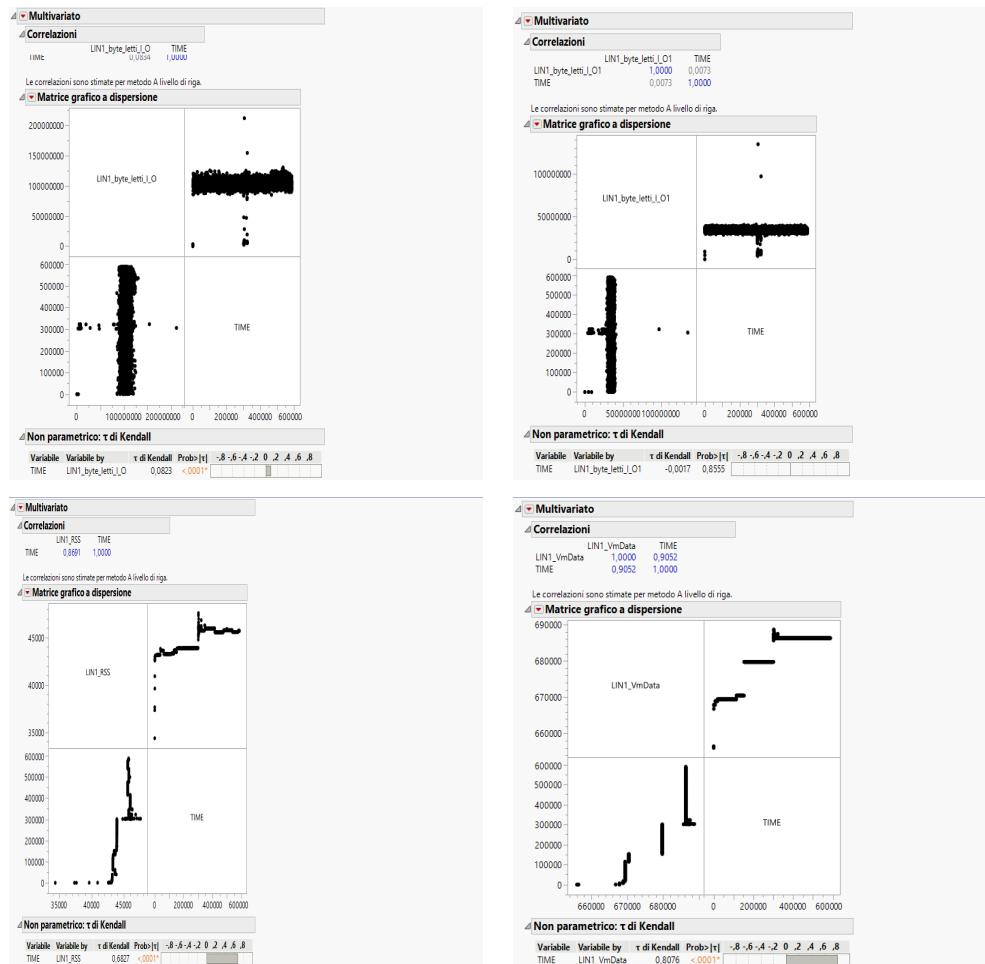


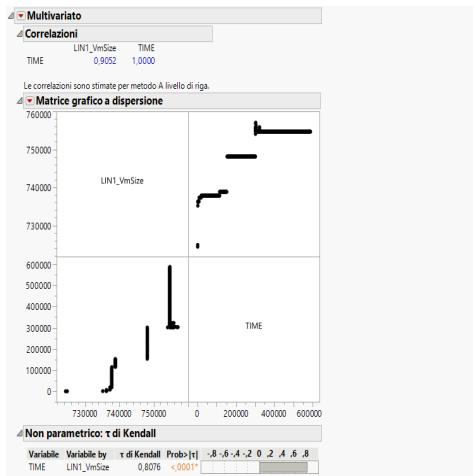
Come modello regressivo non parametrico è stato scelto il test di Mann-Kendall, che non richiede che i dati siano normalmente distribuiti o lineari.

In questo caso, il test per la variabile Byte letti IO1 ha dato come risultato una τ molto vicina allo 0 e p-value minori di 0.05, negando quindi l'esistenza

di un trend. Invece per quanto riguarda VmSize, VmData e RSSm Byte letti IO, si è ottenuto un p-value basso che suggerisce la presenza di un trend, molto lieve nel caso dell'ultima variabile, che presenta un valore di τ basso, più intenso per le altre tre.

Variabile	τ	P	H
Bytes letti IO	0.0823	0.0001	1
Bytes letti IO1	-0.0017	0.855	0
VmSize	0.6827	0.0001	1
VmData	0.8076	0.0001	1
VmRss	0.8076	0.0001	1



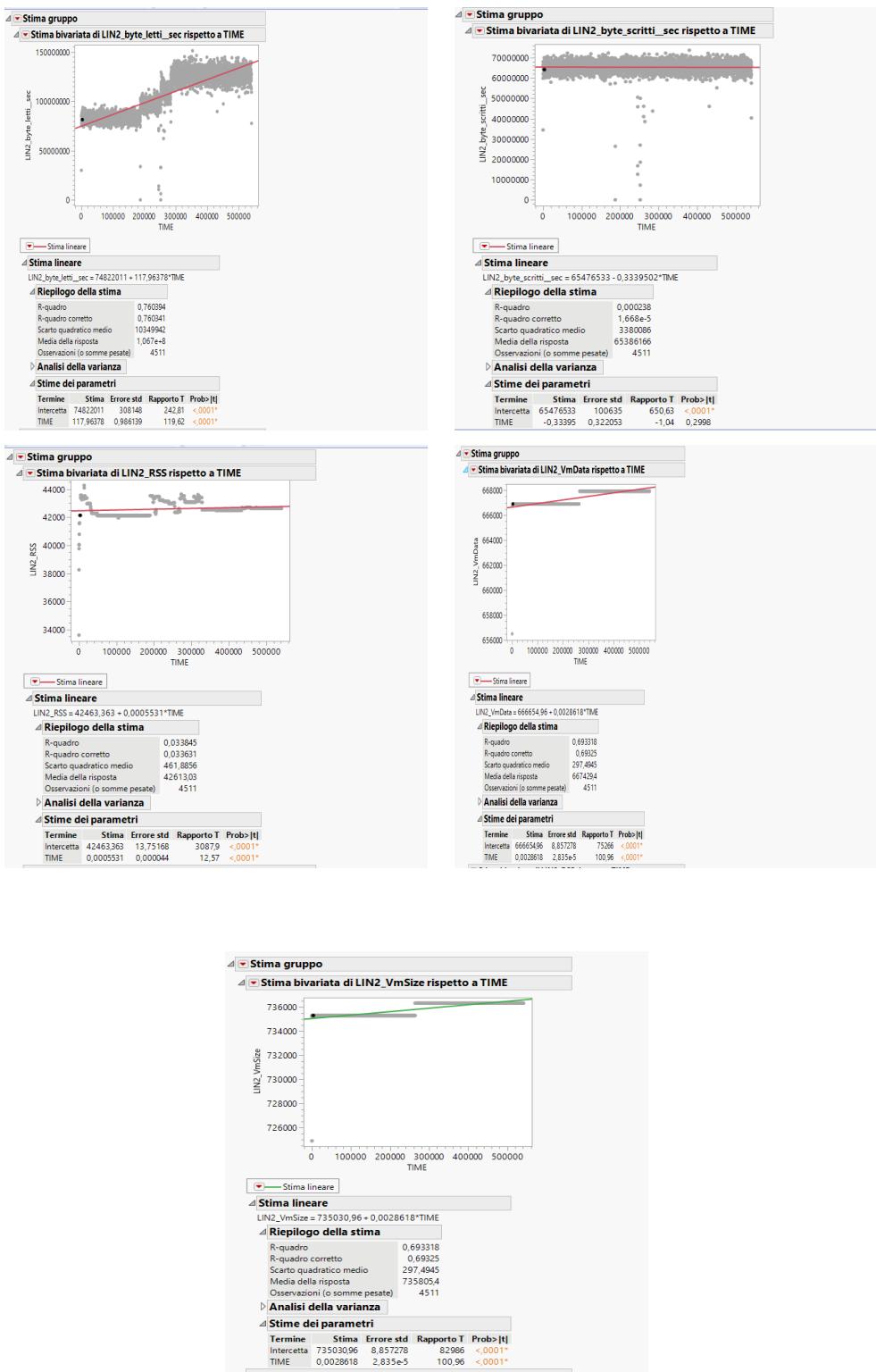


In ultima analisi, è stata utilizzato lo stimateur de Theil-Sen attraverso uno script Python per ottenere una stima di b1.

```
PS C:\Users\litalma\Desktop\Impianti di Elaborazione\Homework\Regressione> python .\sensprocedure.py .\Homework_Regression_2022.xls 2 TIME LIN1_VmSize
Slope: 0.039715052172645887, Interval: [0.032320997662647015] Intercept: 738854.3947801726
PS C:\Users\litalma\Desktop\Impianti di Elaborazione\Homework\Regressione> python .\sensprocedure.py .\Homework_Regression_2022.xls 2 TIME LIN1_VmData
Slope: 0.039715052172645887, Interval: [0.032320997662647015] Intercept: 670464.3947801726
PS C:\Users\litalma\Desktop\Impianti di Elaborazione\Homework\Regressione> python .\sensprocedure.py .\Homework_Regression_2022.xls 2 TIME LIN1_RSS
Slope: 0.084970766233918129, Interval: [0.084876640454962788, 0.085055788865778881] Intercept: 42428.88701754386
PS C:\Users\litalma\Desktop\Impianti di Elaborazione\Homework\Regressione> python .\sensprocedure.py .\Homework_Regression_2022.xls 2 TIME LIN1_BytE_l
Slope: 0.737978125, Interval: [3.6739155483258776, 5.08459886547812] Intercept: 182763229.5673125
```

Secondo dataset E' stato applicato un modello regressivo lineare, per cui la risposta è una funzione lineare del predittore. Infatti, come si può notare dalle figure, sono state eseguite le stime lineari delle variabili VmSize, VmData, RSS, Byte letti e Byte scritti rispetto alla variabile di predizione time.

Nel riepilogo della stima viene indicato l'R-quadro, ovvero il coefficiente di determinazione, che misura il legame tra la variabilità dei dati e la correttezza del modello statistico utilizzato. Nei 5 casi considerati, l'R-quadro assume un valore molto basso per le variabili Byte scritti e RSS e alto per le altre 3 variabili, che quindi vengono stimate abbastanza bene con un modello regressivo lineare.

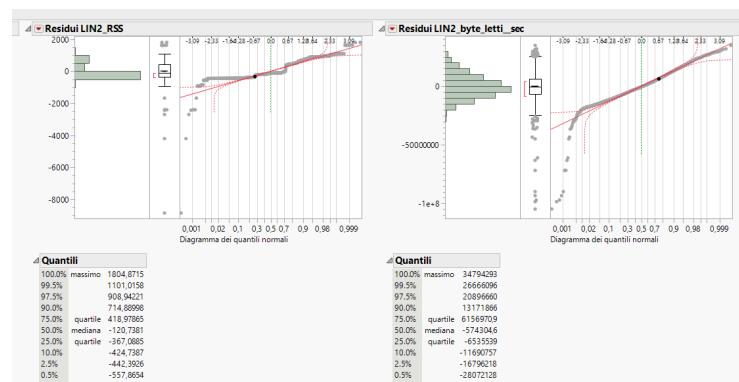
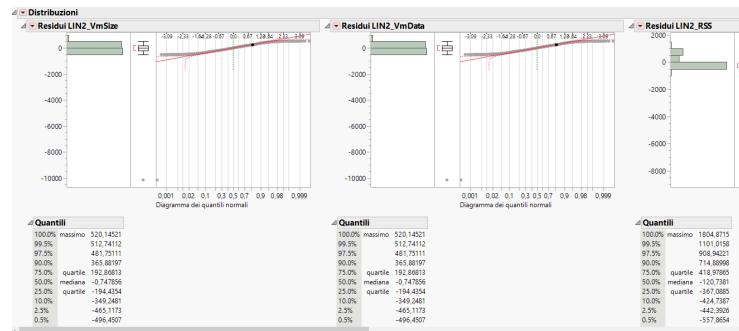


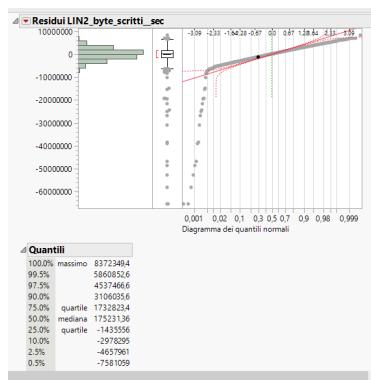
Successivamente, sono state analizzate le distribuzioni dei residui di VmSize, VmData, RSS, Byte letti e Byte scritti. In tutti e 5 i casi, tramite un test

visivo del QQ-plot, è stato verificato che le distribuzioni fossero non normali.

Di conseguenza, è stato applicato un modello non parametrico.

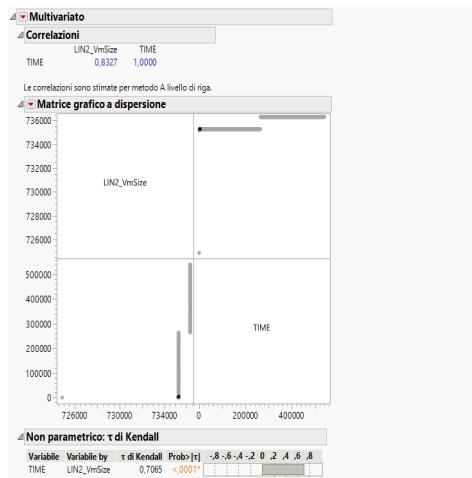
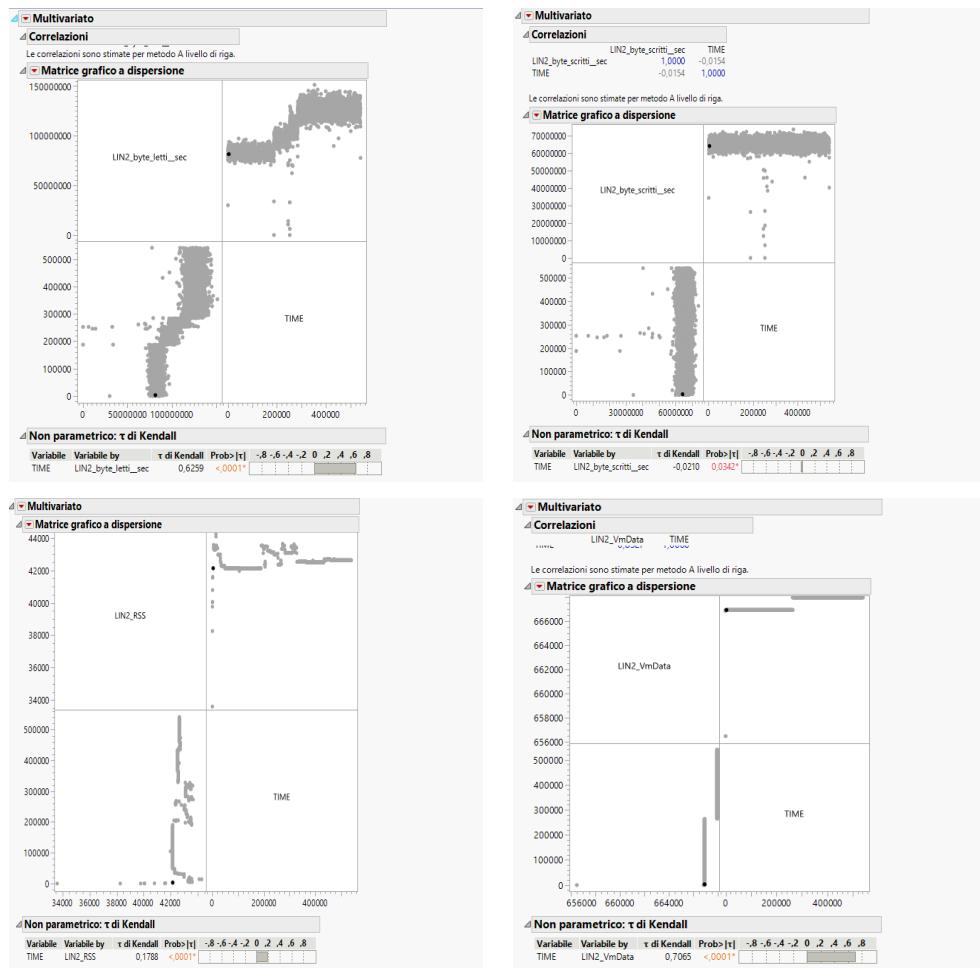
Variabile	τ	P	H
VmData	0.7832	0.0001	1
VmSize	0.7832	0.0001	1
RSS	0.5938	0.0001	1
Byte letti	-0.0506	0.0001	1
Byte scritti	-0.0171	0.0841	0





Come modello regressivo non parametrico è stato scelto il test di Mann-Kendall, che non richiede che i dati siano normalmente distribuiti o lineari.

In questo caso, i test di Byte scritti e RSS hanno dato come risultato una τ molto vicina allo 0 e p-value minori di 0.05, negando quindi l'esistenza di un trend. In particolare, per RSS c'è un valore limite con una τ pari a 0,1: in questo contesto, considerando anche le stime precedenti, si può affermare che l'ipotesi nulla non viene rigettata e quindi c'è un trend. Invece per quanto riguarda VmSize, VmData e Byte letti si è ottenuta una τ vicina ad 1, che, anche grazie ad un p-value alto, permette di verificare l'esistenza di un trend sui dati.



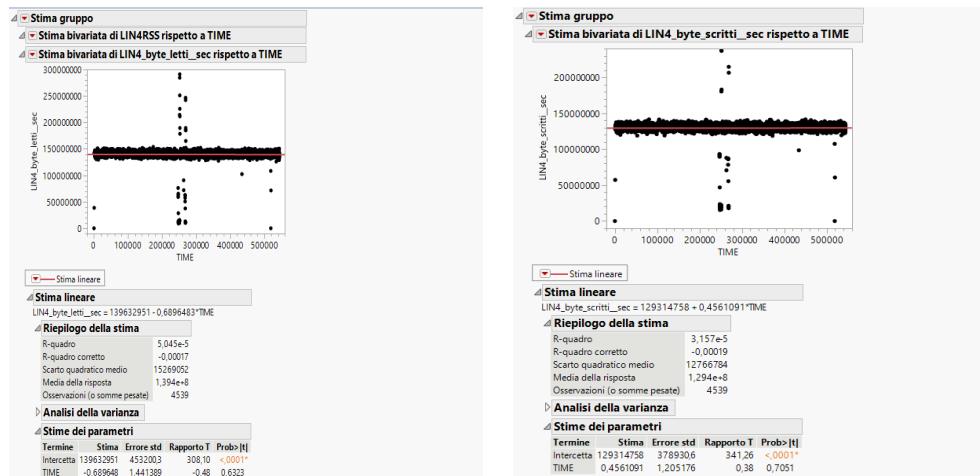
In ultima analisi, è stata utilizzato lo stimatore di Theil–Sen attraverso uno script Python per ottenere una stima di b_1 . Si osservi che la stima è precisa

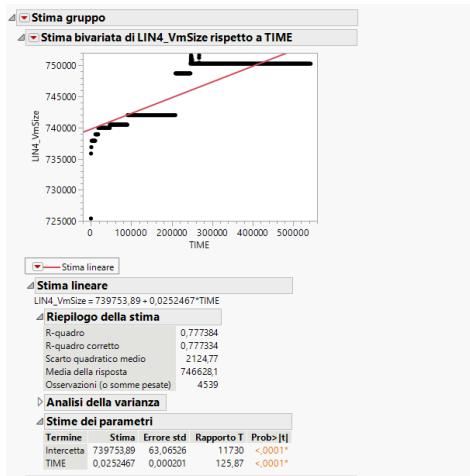
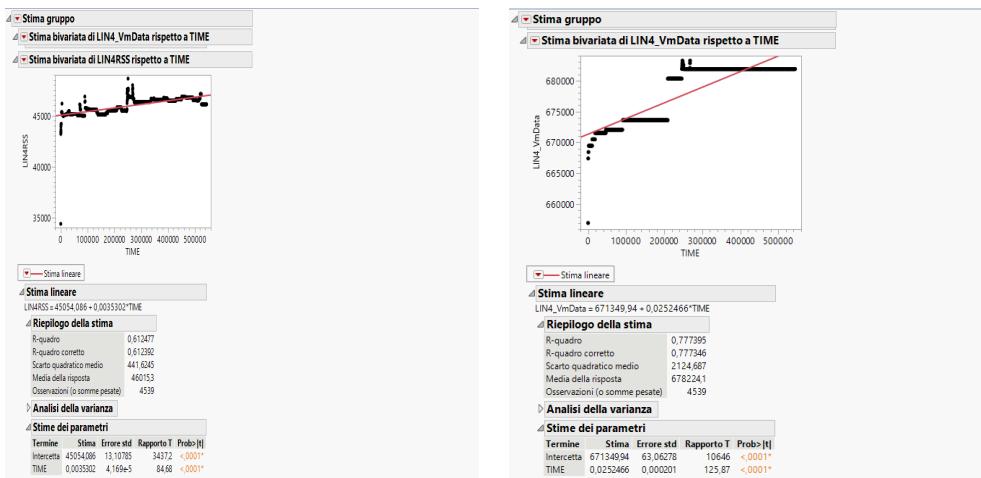
per i primi 3 casi, per gli ultimi 2 invece c'è una correttezza solo nell'ordine di grandezza e nel segno ma non è precisa come per le prime 3 variabili.

```
PS C:\Users\litalma\Desktop\Impianti di Elaborazione\Homework> python \sensprocedure.py \Homework_Regression_2022.xls 3 TIME LIN2_VmSize
Slope: 0.019128747423290214, Interval: [0, 0, 0, 0.00288369827385311] Intercept: 735798.37614880455
PS C:\Users\litalma\Desktop\Impianti di Elaborazione\Homework> python \sensprocedure.py \Homework_Regression_2022.xls 3 TIME LIN2_VmData
Slope: 0.019128747423290214, Interval: [0, 0, 0, 0.00288369827385311] Intercept: 667014.37614880455
PS C:\Users\litalma\Desktop\Impianti di Elaborazione\Homework> python \sensprocedure.py \Homework_Regression_2022.xls 3 TIME LIN2_RSS
Slope: 0.0006324666198172874, Interval: [0, 0.000555555555555556, 0, 0.00074042675481386] Intercept: 42384.854532697744
PS C:\Users\litalma\Desktop\Impianti di Elaborazione\Homework> python \sensprocedure.py \Homework_Regression_2022.xls 3 TIME LIN2_byte_l
etts_sec
Slope: 111.55358983796296, Interval: [112, 73359327217125, 116, 37659764826176] Intercept: 77733384.23786722
```

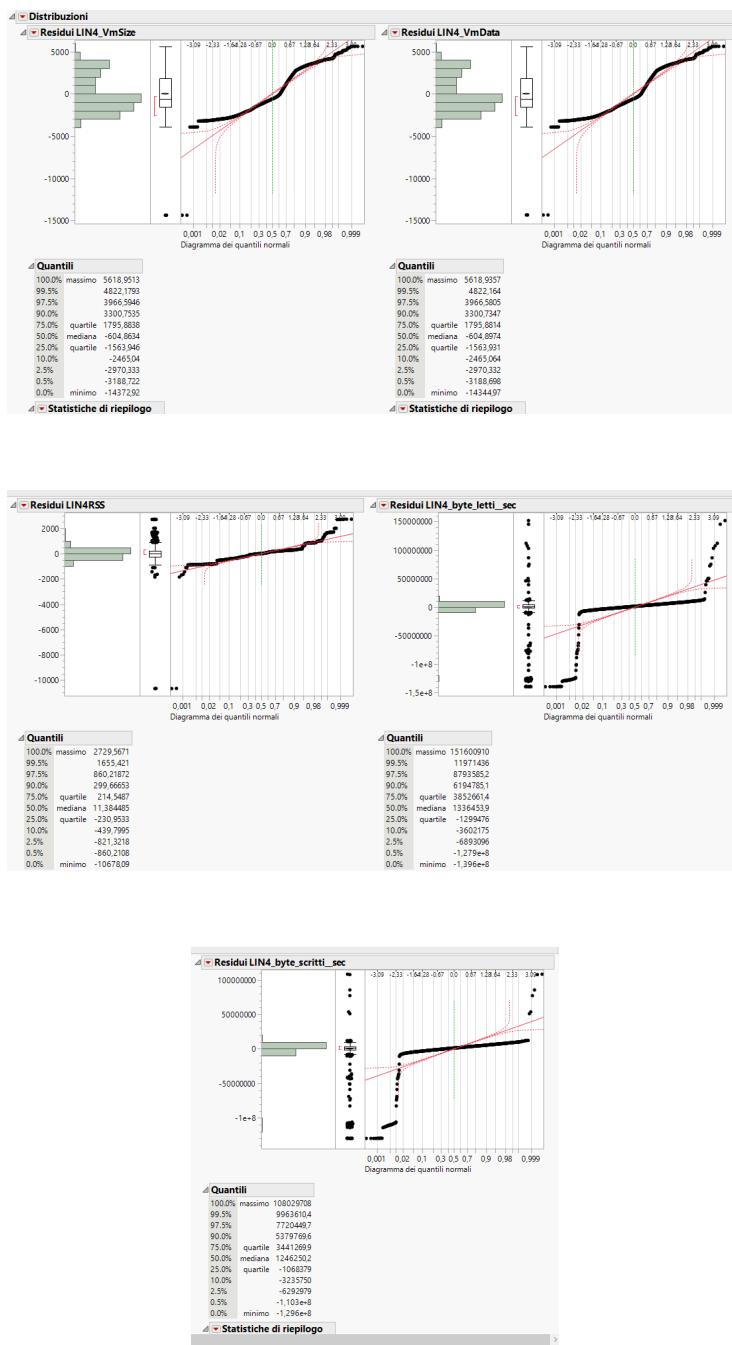
Terzo dataset E' stato applicato un modello regressivo lineare, per cui la risposta è una funzione lineare del predittore. Infatti, come si può notare dalle figure, sono state eseguite le stime lineari delle variabili VmSize, VmData, RSS, Byte letti e Byte scritti rispetto alla variabile di predizione time.

Nel riepilogo della stima viene indicato l'R-quadro, ovvero il coefficiente di determinazione, che misura il legame tra la variabilità dei dati e la correttezza del modello statistico utilizzato. Nei 5 casi considerati, l'R-quadro assume un valore molto basso per le variabili Byte scritti e Byte letti e alto per le altre 3 variabili, che quindi vengono stimate abbastanza bene con un modello regressivo lineare.





Successivamente, sono state analizzate le distribuzioni dei residui di VmSize, VmData, RSS, Byte letti e Byte scritti. In tutti e 5 i casi, tramite un test visivo del QQ-plot, è stato verificato che le distribuzioni fossero non normali. Di conseguenza, è stato applicato un modello non parametrico.

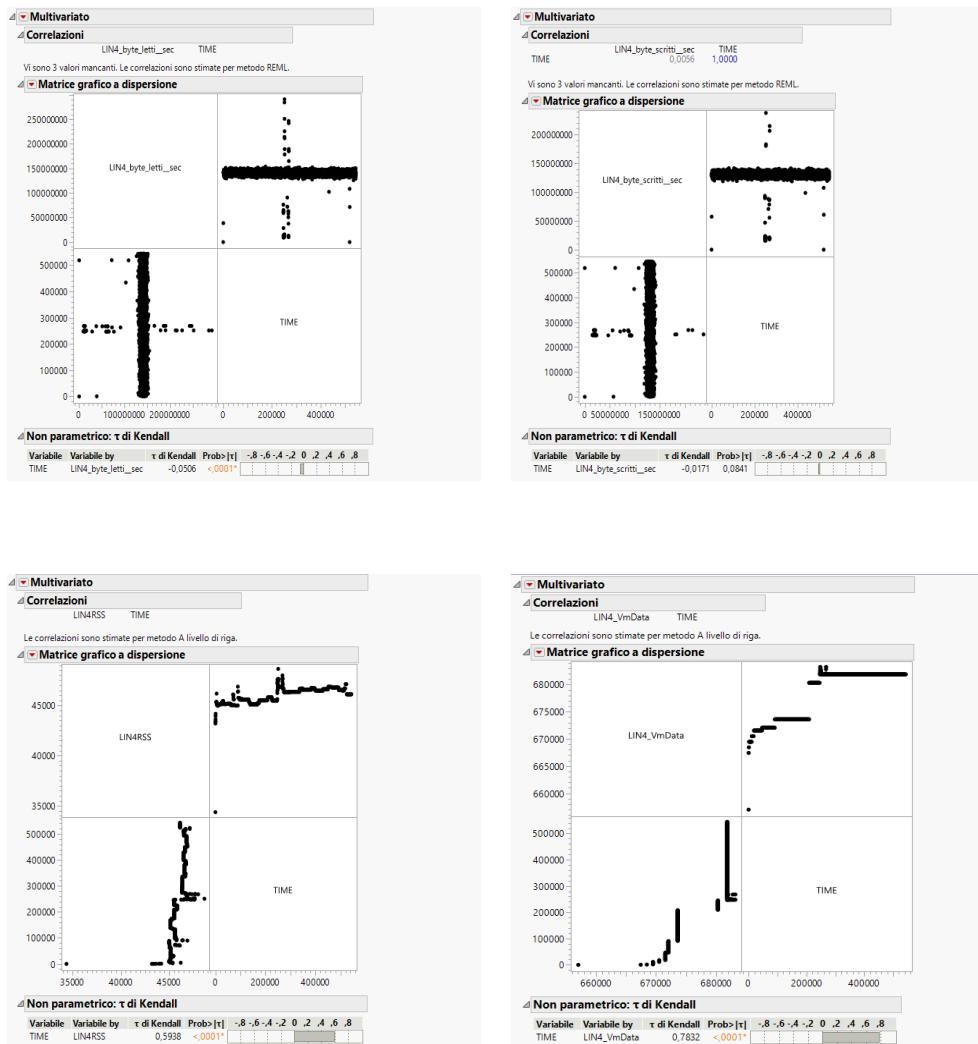


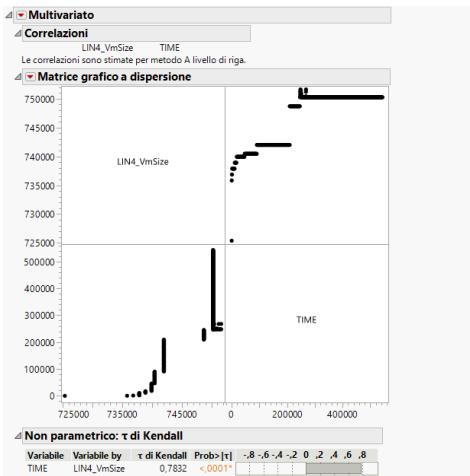
Come modello regressivo non parametrico è stato scelto il test di Mann-Kendall, che non richiede che i dati siano normalmente distribuiti o lineari.

In questo caso, i test di Byte scritti e Byte letti hanno dato come risultato una τ molto vicina allo 0 e p-value minori di 0.05, negando quindi l'esistenza di un trend. Invece per quanto riguarda VmSize, VmData e RSS si è ottenuta

una τ vicina ad 1, che, anche grazie ad un p-value alto, permette di verificare l'esistenza di un trend sui dati.

Variabile	τ	P	H
Bytes letti	-0.0506	0.0001	1
Bytes Scritti	-0.00171	0.0841	0
RSS	0.5938	0.0001	1
VmData	0.7832	0.0001	1
VmSize	0.7832	0.0001	1





In ultima analisi, è stata utilizzato lo stimatore di Theil–Sen attraverso uno script Python per ottenere una stima di b_1 . Anche per questo dataset, la stima qui effettuata risulta essere in linea con i risultati trovati con la stima lineare unicamente per i primi 3 casi.

```
Ps C:\Users\itama\Desktop\Impianti di elaborazione\Homework\Regressione> python .\sensprocedure.py \Homework_Regression_2022.xls 4 TIME LIN4_VmData
Slope: 0.821659324022776844. Interval: [0.82127360228293, 0.8220266666666666]
Intercept: nan
Ps C:\Users\itama\Desktop\Impianti di elaborazione\Homework\Regressione> python .\sensprocedure.py \Homework_Regression_2022.xls 4 TIME LIN4_VmSize
Slope: 0.821659324022776844. Interval: [0.82127360228293, 0.8220266666666666]
Intercept: nan
Ps C:\Users\itama\Desktop\Impianti di elaborazione\Homework\Regressione> python .\sensprocedure.py \Homework_Regression_2022.xls 4 TIME LIN4_RSS
Slope: -0.891848815359477. Interval: [-0.8933883947479881405, -0.8833223947783029404]
Intercept: nan
ps C:\Users\itama\Desktop\Impianti di elaborazione\Homework\Regressione> python .\sensprocedure.py \Homework_Regression_2022.xls 4 TIME LIN4_byte_l
ctti_sec
Slope: -1.891848815359477. Interval: [-2.618383689864559, -1.18788488582996]
Intercept: nan
```

4.4 Esercizio 4 - Predizione

In questo esercizio vengono forniti 3 dataset che tengono traccia della quantità di memoria heap allocata nel tempo e su cui è richiesto di effettuare failure prediction.

Innanzitutto, bisogna rilevare un trend di consumo dell'heap e, qualora fosse rilevato, è necessario stimare il tempo in cui l'heap satura, ovvero supera il limite massimo di 1GByte.

Primo dataset È stato applicato un modello regressivo lineare: infatti, come si può notare dalla figura 4.1, è stata eseguita la stima lineare della variabile allocated heap rispetto alla variabile di predizione Ts. L'R-quadro però risulta troppo basso quindi la stima lineare non può essere applicata a

questo esercizio.

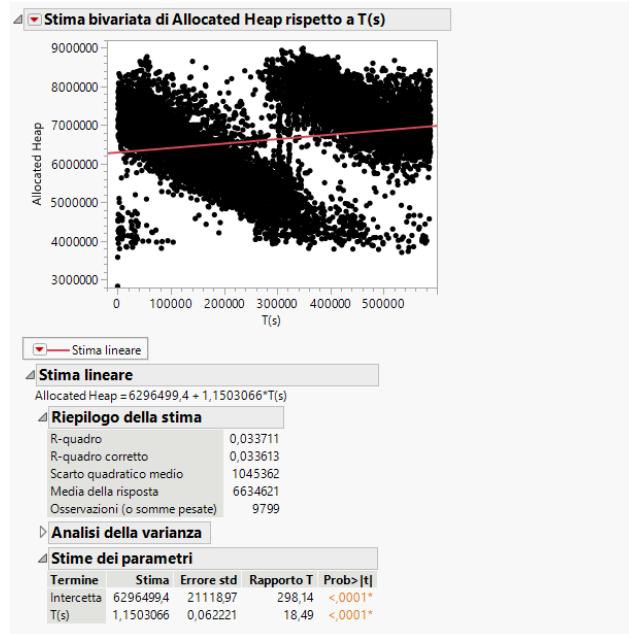
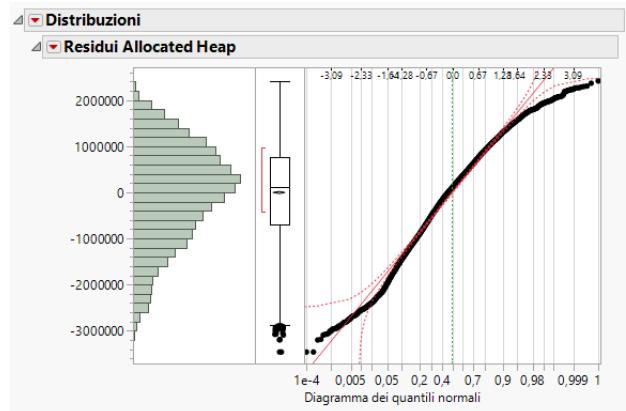


Figura 4.1: Stima lineare primo dataset

E' stata quindi verificata la normalità dei residui, attraverso un test visivo:



Dal momento che i residui hanno una distribuzione non normale, è stato utilizzato lo stimatore di Theil-Sen per stimare la slope e l'intercetta della retta:

```
PS C:\Users\itama\Desktop\Impianti di Elaborazione\Homework\Regressione> python .\snsprocedure.py .\HomeWork_Regression_2022.xls 5 Ts Allocated_Heap
p
slope: 0.6911224682945296, interval: [0.563777264562484, 0.8178240889963452], Intercept: 6562275.461669586
```

La stima è la seguente:

$$allocated\ heap = 6562275,46 + (0,69 \pm 0,12) * T(s)$$

Quindi:

$$T(s) = \frac{1000000000 - 6562275,46}{(0,69 \pm 0,12)} \approx 46 \pm 8 \text{ anni}$$

Secondo dataset E' stato applicato un modello regressivo lineare: infatti, come si può notare dalla figura 4.2, è stata eseguita la stima lineare della variabile allocated heap rispetto alla variabile di predizione Ts. L'R-quadro però risulta troppo basso quindi la stima lineare non può essere applicata a questo esercizio.

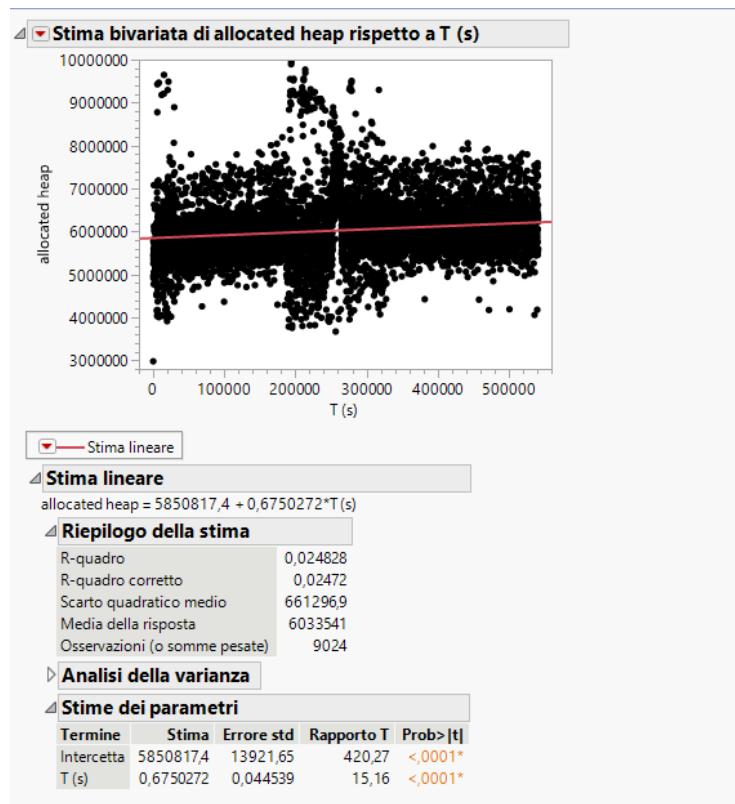
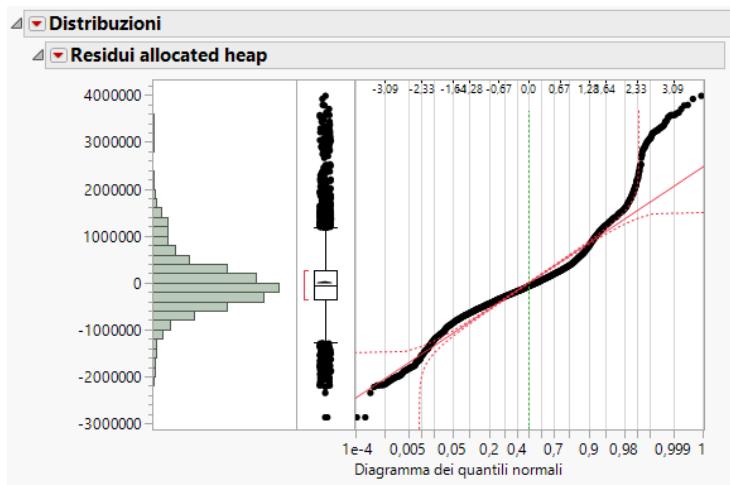


Figura 4.2: Stima lineare secondo dataset

E' stata quindi verificata la normalità dei residui, attraverso un test visivo:



Dal momento che i residui hanno una distribuzione non normale, è stato utilizzato lo stimatore di Theil-Sen per stimare la slope e l'intercetta della retta:

```
ps C:\Users\itama\Desktop\Impianti di Elaborazione\Homework\Regressione> python .\sensprocedure.py .\HomeWork_Regressien_2022.xls 6 Ts allocated_heap
p
Slope: 0.678094993804572173, Interval: [0.6648638015610893, 0.7366538131962297] Intercept: 5780180.563324536
```

La stima è la seguente:

$$\text{allocatedheap} = 5780180,56 + (0,67 \pm 0,6) * T(s)$$

Quindi:

$$T(s) = \frac{1000000000 - 5780180,56}{(0,67 \pm 0,6)} \approx 41 \pm 10 \text{ anni}$$

Terzo dataset E' stato applicato un modello regressivo lineare: infatti, come si può notare dalla figura 4.3, è stata eseguita la stima lineare della variabile allocated heap rispetto alla variabile di predizione Ts. L'R-quadro però risulta troppo basso quindi la stima lineare non può essere applicata a questo esercizio.

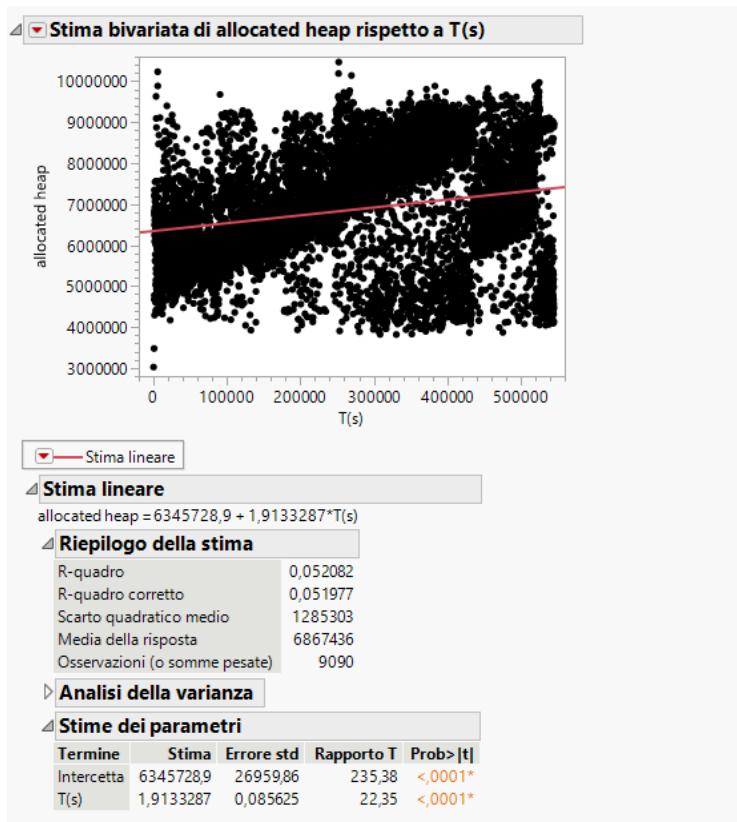
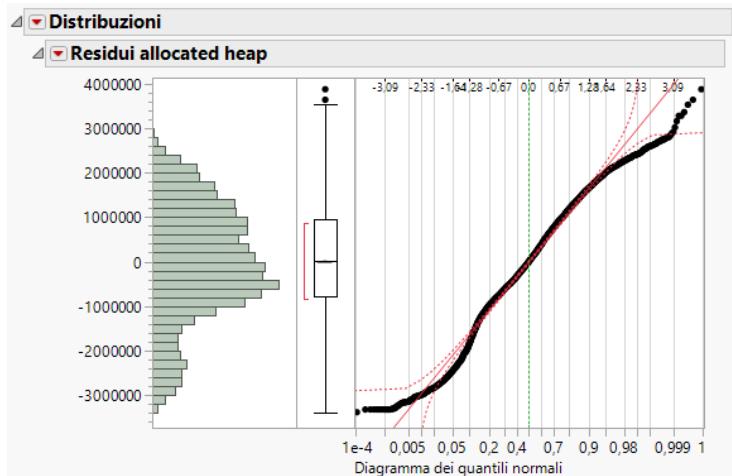


Figura 4.3: Stima lineare terzo dataset

E' stata quindi verificata la normalità dei residui, attraverso un test visivo:



Dal momento che i residui hanno una distribuzione non normale, è stato utilizzato lo stimatore di Theil-Sen per stimare la slope e l'intercetta della retta:

```
PS C:\Users\itama\Desktop\Impianti di Elaborazione\Homework\Regressione> python .\sensprocedure.py .\HomeWork_Regression_2022.xls 7 Ts allocated_heap
p
Slope: 2.9638932760062105, interval: [2.714759535655058, 3.0923529411764785] Intercept: 6041585.556431387
```

La stima è la seguente:

$$allocated\text{heap} = 6041585,56 + (2,9 \pm 0,19) * T(s)$$

Quindi:

$$T(s) = \frac{1000000000 - 6041585,56}{(2,9 \pm 0,19)} \approx 20 \pm 10 \text{ anni}$$

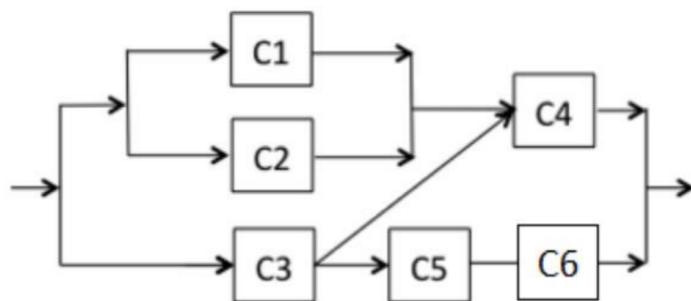
Capitolo 5

Reliability

In questa sezione sono svolti ed analizzati alcuni esercizi che richiedono l'analisi di reliability di alcuni sistemi di esempio tramite l'utilizzo di tecniche model-based analitiche.

5.1 Esercizio 1 - RBD

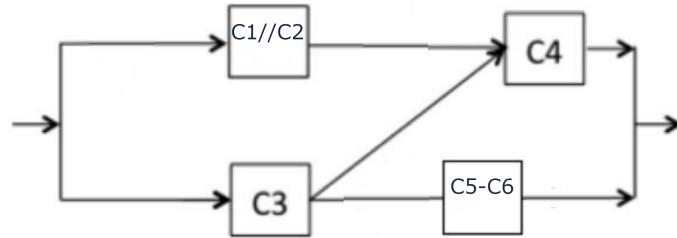
Il sistema sotto analisi è composto da sei sottosistemi le cui proprietà di reliability sono note, tutti i blocchi hanno infatti la stessa reliability pari a $R(t) = e^{-\lambda t}$,



In prima analisi si evince immediatamente la relazione di parallelismo presente tra i sottosistemi C1 e C2 e quella di serie tra i sistemi C5 e C6. Le funzioni di reliability per i blocchi sono quindi, in accordo con le formule standard di riduzione serie e parallelo:

$$R_5 - R_6 = R_5 * R_6$$

$$R_1 // R_2 = 1 - (1 - R_1)(1 - R_2)$$



Per l'analisi non è sufficiente l'utilizzo di tecniche standard di riduzione serie parallelo, ma è necessario operare sfruttando il condizionamento.

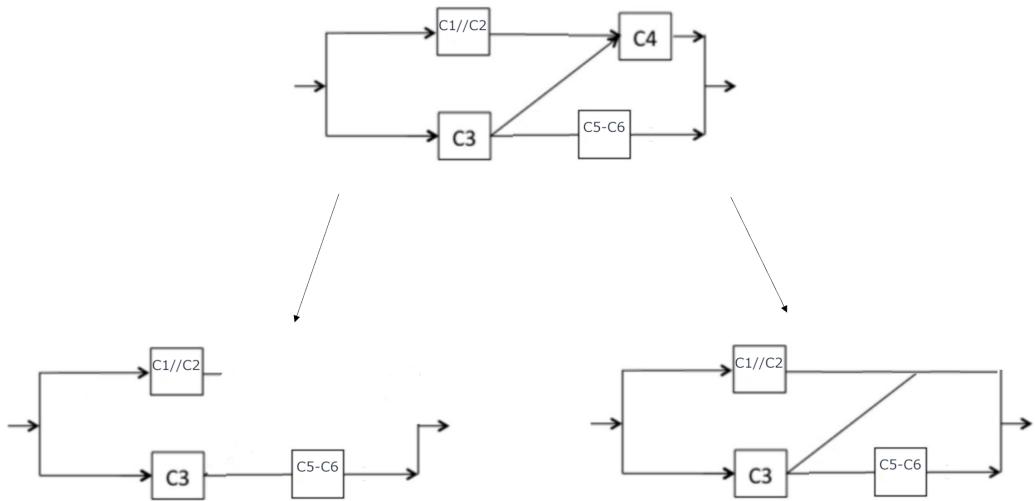
Prima di procedere all'analisi corretta della reliability abbiamo calcolato un upper-bound con la tecnica dei success paths. I success path individuati sono quindi:

- $C_1 - C_4$
- $C_2 - C_4$
- $C_3 - C_4$
- $C_3 - C_5 - C_6$

Otteniamo l'upper bound come parallelo dei success path identificati:

$$R_{sys} \leq 1 - (1 - R^2)^3(1 - R^3) = \dots$$

Procediamo a questo punto all'analisi esatta. Abbiamo scelto di procedere al condizionamento rispetto al sottosistema C_4 .



Una volta effettuata la decomposizione il sistema può essere risolto banalmente. Condizionando il sistema al fallimento del blocco C4, il sistema C3 è in serie al sistema C5-C6. Una volta considerato invece il non fallimento di questo, il sistema può essere risolto come un parallelo tra C1//C2 e C3.

$$P(\text{System}^{UP}) = P(S^{UP}|C_4^{UP}) * P(C_4^{UP}) + P(S^{UP}|C_4^{DOWN}) * P(C_4^{DOWN})$$

Calcoliamo quindi la reliability complessiva del sistema:

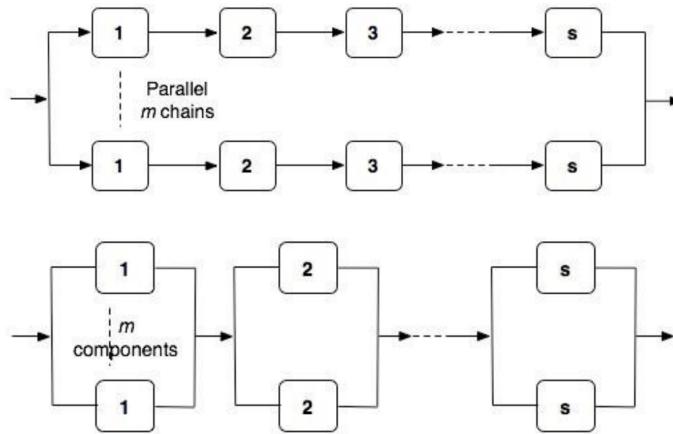
$$R_{sys} = R[1 - (1-R)^3] + (1-R)R^3 = e^{-\lambda t}[1 - (1-e^{-\lambda t})^3 + (1-e^{-\lambda t})e^{-3\lambda t}] = -2e^{-3\lambda t} + 3e^{-2\lambda t}$$

Calcoliamo quindi il MTTF:

$$MTTF_{sys} = \int_0^\infty R_{sys} = \frac{5}{6\lambda}$$

5.2 Esercizio 2 - Confronto tra sistemi

Il secondo esercizio richiede di confrontare la reliability di due sistemi composti dagli stessi sottosistemi, ma con una topologia differente.



Da una prima analisi qualitativa ci rendiamo subito conto che il secondo sistema è più reliable del primo, il numero dei success path è infatti notevolmente maggiore.

Il primo sistema è composto dal parallelo di m catene composte dalla serie di s sottosistemi dalle stesse proprietà. La reliability di ogni singola catena può essere facilmente calcolata ed è pari a:

$$R_{chain} = R^s$$

Otteniamo quindi che, la reliability del parallelo di m catene è pari a:

$$R_{sys_1} = [1 - (1 - R^s)^m]$$

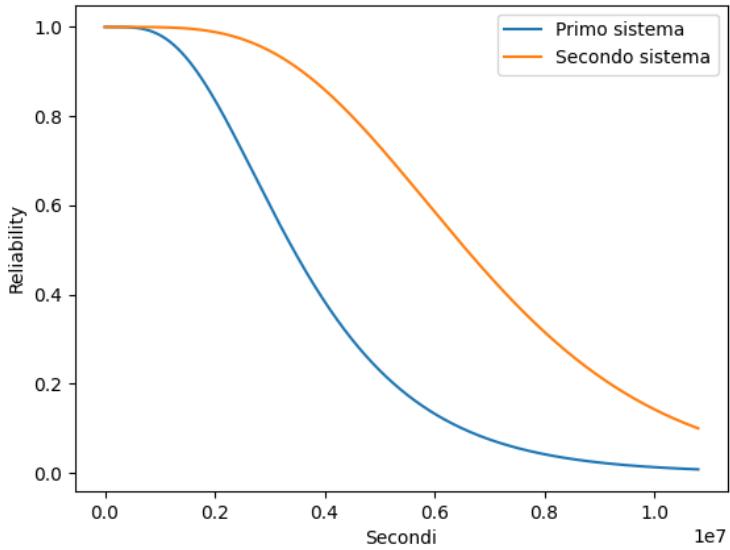
Il secondo è invece costituito dalla serie di s paralleli di m sottosistemi. Ogni parallelo ha reliability pari a:

$$R_p = 1 - (1 - R)^m$$

Otteniamo quindi che la reliability del secondo sistema è pari a:

$$R_{sys_2} = [1 - (1 - R)^m]^s$$

Particularizzando per i valori di $s = 3$ ed $m = 5$, e considerando, inoltre, $R = e^{-\lambda t}$, con $\lambda = \frac{1}{1400h}$, possiamo procedere ad un confronto grafico tra le reliability dei due sistemi al variare del mission time:



Da questa analisi notiamo quindi come, a prescindere dal valore di mission time, il secondo sistema risulta drasticamente più reliabile del primo, in accordo con l'intuizione iniziale.

L'esercizio richiede poi di modificare il primo sistema allo scopo di garantire, per uno specifico mission time, la stessa reliability del secondo.

Nel nostro caso abbiamo optato per un mission time pari al MTTF dei singoli componenti alla base dei due sistemi, ovvero uguale a 1400 ore.

Per questo specifico mission time la reliability del secondo sistema è pari a:

$$R_{sys_1}(1400h) \simeq 73\%$$

che è un valore molto superiore a quella ottenuta dal primo sistema.

Per garantire la stessa reliability modifichiamo il primo sistema aumentando il numero di catene in parallelo ed introducendo, quindi, maggiore ridondanza.

Partendo dalla formula parametrica calcolata in precedenza che lega la reliability del primo sistema al parametro m pari al numero di catene in parallelo, possiamo facilmente calcolare il valore di m tale da garantire la seguente

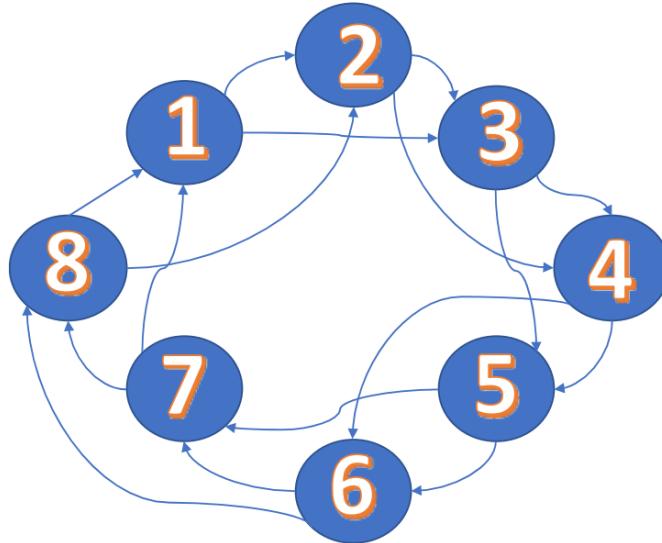
relazione:

$$R_{sys_1}(t = 1400h, m) = [1 - (1 - R^3)^m] = 73\%$$

Il risultato ottenuto è $m \simeq 26$, giungiamo quindi alla conclusione che è necessario aumentare drasticamente la ridondanza del primo sistema per garantire, per questo specifico mission time, la stessa reliability del secondo, che è meglio strutturato.

5.3 Esercizio 3 - Skip ring

L'esercizio richiede di valutare la reliability di un sistema connesso con topologia skip-ring. Questa topologia è costruita in modo tale da garantire che il fallimento di un nodo non limiti la possibilità degli altri nodi di comunicare, ogni nodo è quindi connesso, oltre che ai nodi adiacenti, anche a quelli successivi. La topologia è quindi la seguente:



Per il corretto funzionamento della rete skip-ring è necessario che si verifichi un determinato evento: non devono fallire due nodi adiacenti. Se ciò si verificasse, la rete non sarebbe più in grado di instradare i dati su un collegamento alternativo.

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

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

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

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

- $i=0$: se tutti i nodi funzionano correttamente, banalmente il sistema funziona sempre

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

- $i=1$: se fallisce un solo nodo, il sistema continua a funzionare (è alla base dello scopo della skip-ring network)

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

- $i=2$: se falliscono due nodi, vi sono 8 combinazioni per cui il sistema non funziona, che si verificano quando due nodi adiacenti falliscono

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

- $i=3$: se falliscono tre nodi, vi saranno due casi in cui il sistema non funziona

- falliscono 3 nodi adiacenti: 8 combinazioni possibili
- falliscono 2 nodi adiacenti e un terzo non adiacente: per ogni coppia di nodi adiacenti vi sono 4 nodi non adiacenti che possono fallire, quindi esistono 32 combinazioni possibili di questo tipo

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

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

- i=4: se falliscono 4 nodi, si valutano dualmente le combinazioni che permettono ancora il funzionamento del sistema. Nelle combinazioni possibili i nodi falliti si alternano con i funzionanti, ovvero:

- 1-3-5-7 funzionano, 2-4-6-8 funzionano;
- 2-4-6-8 funzionano, 1-3-5-7 non funzionano;

Esistono quindi soltanto 2 combinazioni in cui il sistema funziona:

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

Infine, per i maggiore di 4, non è presente alcuna combinazione che permetta il funzionamento del sistema, avremo sempre almeno una coppia di nodi adiacenti down.

Di conseguenza, supponendo che la reliability del singolo componente segua la legge di fallimento esponenziale con un failure rate pari a $\lambda = 0.005 \text{ failure}/h$, è possibile riscrivere R_m nel seguente modo:

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

Si può quindi ottenere la reliability della skip-ring network sommando i vari contributi ottenuti al variare di i :

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

Possiamo quindi andare a calcolare il valore numerico della reliability per un mission time di 48h:

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

5.4 Esercizio 4 - RBD

Per lo svolgimento di questo esercizio è richiesto di confrontare la reliability di 4 coppie di sistemi composti dagli stessi sottosistemi A, B, C , caratterizzati dalle seguenti proprietà di reliability:

- Distribuzione dei failure esponenziale
- $MTTF_A = 900h$
- $MTTF_B = 7000h$
- $MTTF_C = 1000h$

La prima coppia di sistemi da confrontare è la seguente:



Da un'analisi qualitativa, il primo sistema risulta essere più reliable: il fallimento di un sottosistema a, nel primo caso, non comporta automaticamente il fallimento del sistema, visto che il componente è replicato; nel secondo sistema, invece, non è presente la ridondanza del componente a.

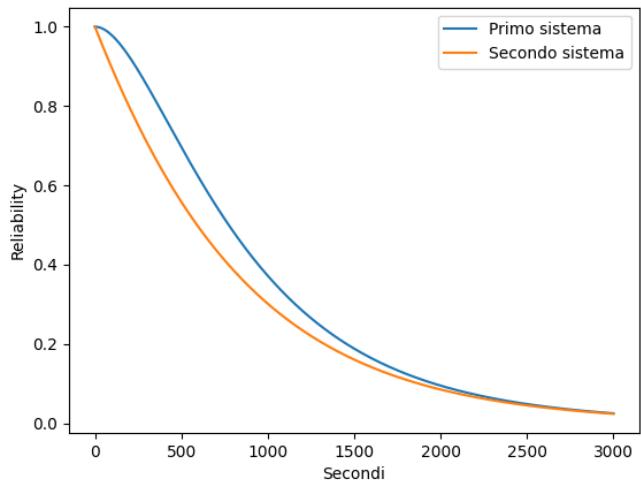
Il primo sistema è costituito dal parallelo di due serie, la reliability quindi è pari a:

$$R_1 = 1 - (1 - R_a * R_b)(1 - R_a * R_c)$$

Nel secondo caso invece il sistema è composto dalla serie tra A ed il parallelo tra B e C, la reliability è quindi pari a:

$$R_2 = R_a(1 - (1 - R_b)(1 - R_c))$$

Particularizzando i valori delle reliability dei sottosistemi a $R = e^{-\lambda t}$, otteniamo le seguenti funzioni di reliability:



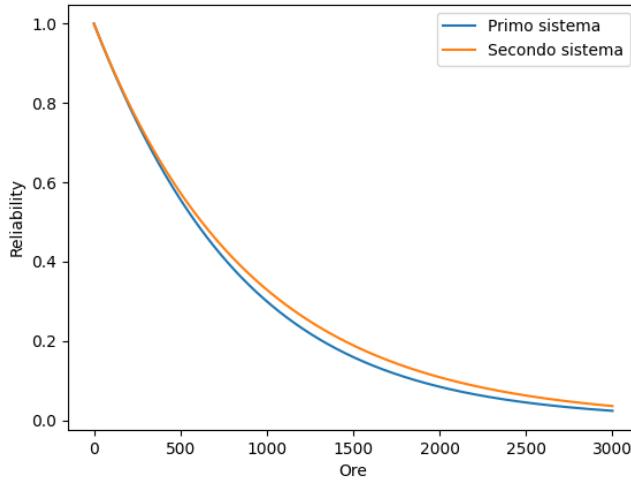
La seconda coppia di sistemi da confrontare è la seguente:



Mentre la reliability del secondo sistema è proprio $R_2 = R_a$, la prima può essere calcolata come serie tra A e il parallelo di A e B :

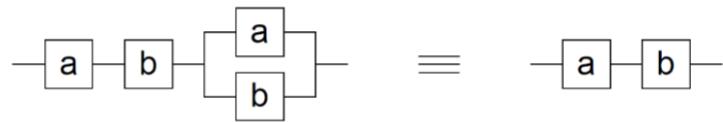
$$R_1 = R_a(1 - (1 - R_a)(1 - R_b))$$

Confrontando le due reliability si ottiene il seguente risultato:



La reliability del secondo sistema è leggermente migliore di quella del primo, che presenta infatti un sottosistema in più in serie. Il risultato era prevedibile, la reliability della serie di più elementi è infatti sempre minore o uguale della reliability dell'elemento meno affidabile nella serie.

La terza coppia è la seguente:



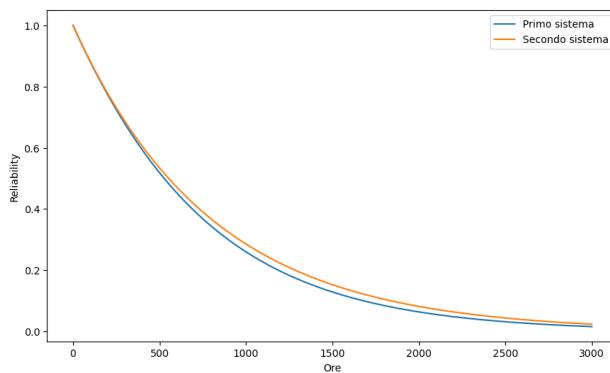
Il ragionamento precedente regge anche in questo caso, la reliability del secondo sistema sarà sicuramente superiore a quella del primo poichè questo presenta un sottosistema, in questo caso il parallelo tra A e B in serie in più.

Analiticamente otteniamo che:

$$R_1 = R_a R_b (1 - (1 - R_a)(1 - R_b))$$

$$R_2 = R_a R_b$$

Confrontando graficamente le due funzioni di reliability otteniamo che il primo sistema è leggermente meno reliable del secondo:



L'ultima coppia è la seguente:

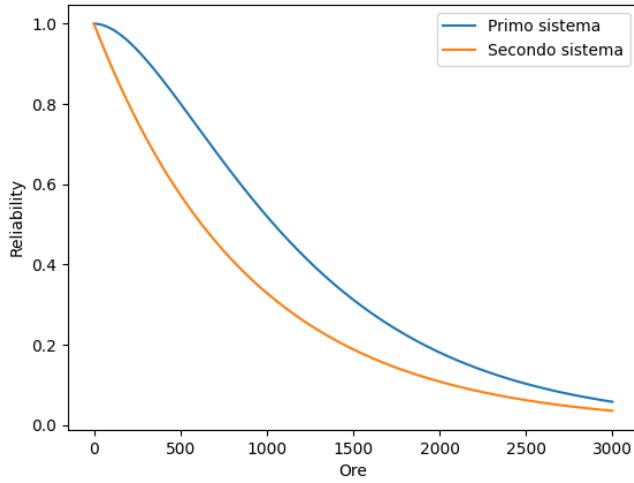


Il primo sistema risulta più reliable, poichè introduce ridondanza addizionale al secondo. Analiticamente si ha infatti che:

$$R_1 = 1 - (1 - R_a)(1 - R_a R_b)$$

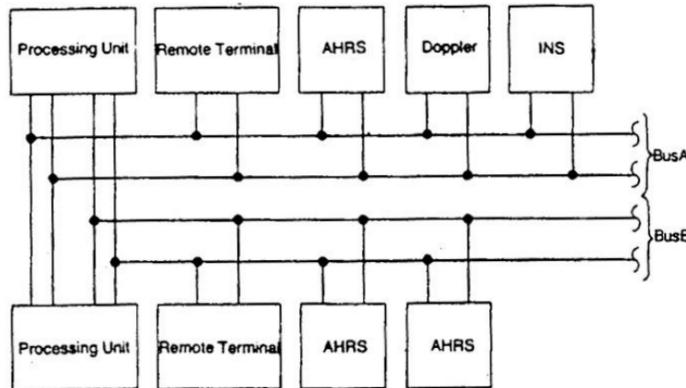
$$R_2 = R_a$$

Confrontando graficamente i due risultati otteniamo come atteso, che il primo sistema è decisamente più reliable del secondo:



5.5 Esercizio 5 - Reliability sistema di pilotaggio

L'ultimo esercizio richiede la modellazione e l'analisi, tramite RBD e FT, della centralina di controllo di un elicottero. Lo schema del dispositivo è il seguente:



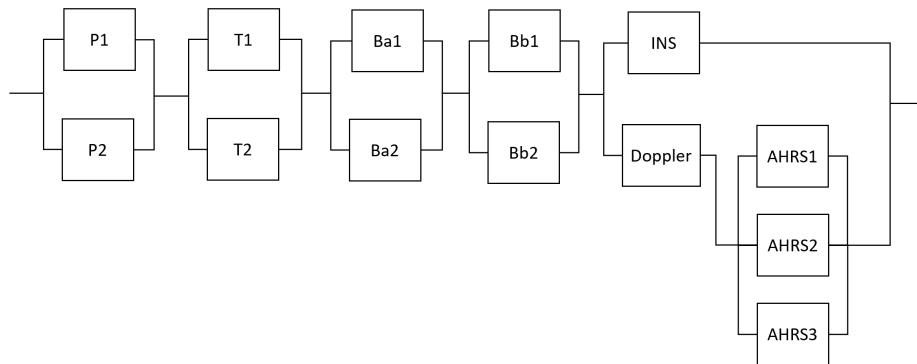
Dallo schema e dalla descrizione fornita dalla traccia giungiamo alle seguenti proprietà fondamentali che caratterizzano il sistema:

- Il sistema presenta due processori ridondanti, ed è in grado di funzionare finché uno dei due funziona

- Il sistema presenta due terminali di controllo ridondanti
- Entrambi i bus utilizzati, bus A e bus B , sono ridondanti
- Il sistema di navigazione è basato sull' Internal Navigation System, ma, per le emergenze, è sostituibile da un Doppler e da uno dei tre moduli AHRS presenti all'interno del sistema.

A partire da questa descrizione possiamo facilmente costruire un reliability block diagram del sistema, avremo infatti che tutti le parti necessarie per il funzionamento del sistema saranno poste in serie mentre i moduli di ridondanza saranno posti in parallelo.

Il risultato della modellazione del sistema è il seguente:

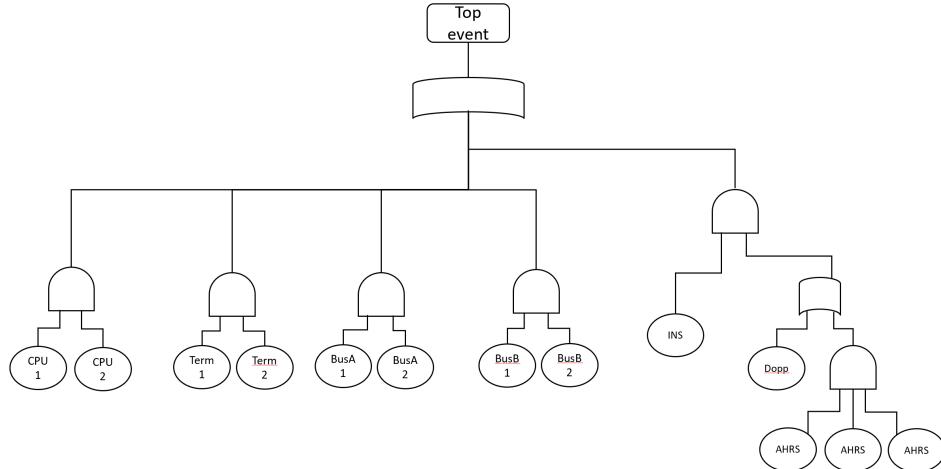


A partire da questo modello possiamo calcolare la reliability complessiva del sistema di volo:

$$\begin{aligned}
 R_{sys} = & [1 - (1 - R_p)^2] * [1 - (1 - R_t)^2] * [1 - (1 - R_{Ba})^2] \\
 & [1 - (1 - R_{Bb})^2] * [1 - (1 - R_{INS})(1 - R_D(1 - (1 - R_{AHRS})^3)] \quad (5.1)
 \end{aligned}$$

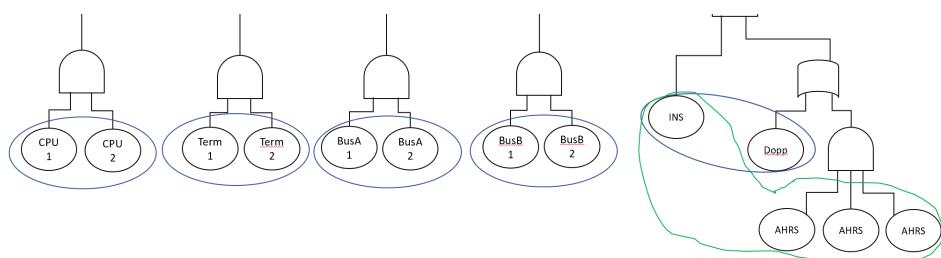
L'esercizio richiede poi di modellare il sistema con l'utilizzo dei fault trees, ed individuare i minimal cutsets che provocano il failure del sistema. Partendo dal RBD è semplice realizzare il FT model, è infatti sufficiente sostituire a

tutte i paralleli delle porte OR e a tutte le serie delle porte AND. Così facendo otteniamo il seguente modello:



A partire da questo modello si calcolano i minimal cut sets, ovvero gli insiemi minimi di sottosistemi il cui malfunzionamento causa il failure del sistema. I minimal cutset individuati sono:

- CPU 1 - CPU 2
- Terminal 1 - Terminal 2
- Bus A 1 - Bus A 2
- Bus B 1 - Bus B 2
- INS - AHRS 1 - AHRS 2 - AHRS 3
- INS - Doppler



A questo punto l'esercizio richiede di calcolare la reliability del sistema, considerando un mission time pari ad $1h$, e fornisce allo scopo la seguente particolarizzazione dei componenti:

Componente	MTTF (hr)
CPU	10000
Remote Terminal	4500
AHRS	2000
INS	2000
Doppler	500
Bus	60000

Tabella 5.1: MTTF dei componenti dell'elicottero.

Noti i $MTTF$ possiamo facilmente calcolare i failure rate λ , e, ipotizzando una distribuzione esponenziale per i fallimenti, otteniamo la reliability dei singoli componenti come $R_i(t) = e^{-\lambda t}$.

Utilizzando la formula precedentemente calcolata e praticolizzata per i valori di reliability dei componenti, otteniamo il seguente valore:

$$R_{sys}(t = 1h) = 0.9999989413227502$$

L'ultima parte dell'esercizio richiede di non assumere il valore di coverage per la fault detection del processore come ideale, ma di calcolare il valore c tale da garantire, per un mission time di 1 ora, una reliability del 99%.

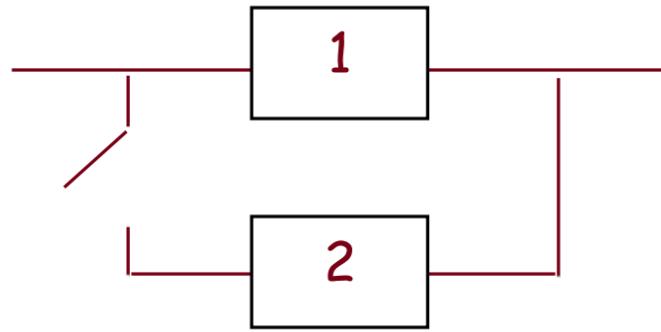


Figura 5.1: Effetto della coverage non ideale sul sistema.

Considerando il fattore di coverage in serie ad uno dei due processori replicati, l'espressione della reliability cambia in questo modo:

$$R_{sys}(t = 1h, C) = [1 - (1 - R_p) * (1 - R_p * C)] * [1 - (1 - R_t)^2] * [1 - (1 - R_{Ba})^2][1 - (1 - R_{Bb})^2] * [1 - (1 - R_{INS})(1 - R_D(1 - (1 - R_{AHRS})^3))] = 0.99$$

Possiamo facilmente da questa espressione ricavare il valore di C necessario, ovvero $C \simeq 0.91$

Capitolo 6

FFDA

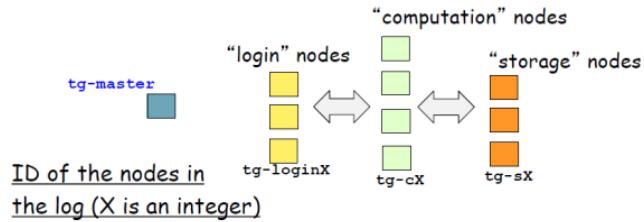
La Field Failure Data Analysis è una tecnica che punta ad analizzare le proprietà di Dependability di un sistema a partire dall'analisi di report e log raccolti in condizioni di operatività del sistema. Questa tecnica measurement-based è senza dubbio estremamente affidabile e consente un'analisi dettagliata e precisa delle proprietà di un sistema, ma allo stesso tempo risulta molto costosa e difficilmente generalizzabile.

L'analisi è effettuata a partire dai log di due supercalcolatori: Mercury e Blue Gene.

6.1 Analisi dei log di Mercury

Il primo file di log, composto da 80854 entry e contenente solo fault fatali del sistema, è prodotto dal sistema Mercury. Si tratta di un supercalcolatore IBM, composto da più nodi di calcolo distinti in cluster.

Ogni cluster rispetta una architettura Three-layered:



I nodi di calcolo presenti nell’architettura sono distinti in:

- Nodo Master: Nodi fondamentali che si occupano della gestione degli altri nodi presenti nel cluster.
- Nodo di login: su cui sono eseguiti servizi di accesso e distribuzione dei job in ingresso al cluster.
- Nodo di calcolo: utilizzati per l’esecuzione di funzionalità cpu-intensive.
- Nodo di memorizzazione: si occupano della memorizzazione dei dati e dei risultati prodotti dai nodi di calcolo.

Ad ogni entry è inoltre associato il nodo del sistema che l’ha generata ed è catalogata in base al componente (IO, MEM, etc) che ha provocato il fallimento. Il log in analisi è stato opportunamente filtrato e pre-elaborato. Le entry presenti sono esclusivamente quelle associate a fault fatali. Ciò semplifica notevolmente la nostra analisi poichè non sono necessarie operazioni di deparametrizzazione e filtraggio.

Il formato di ogni entry presente all’interno del log è il seguente:

- Timestamp
- Originating node
- Originating subsystem
- Text-free message

La prima fase di elaborazione del log ha lo scopo di associare le entry presenti ad effettivi eventi di failure tramite l'utilizzo di una tecnica di coalescenza temporale. Lo scopo è quello di raggruppare più entry del log in tuple, associate ad un evento di failure, su base temporale. La formula utilizzata per il raggruppamento delle entry in tuple è la seguente:

```
IF  $t(X_{i+1}) - t(X_i) < W$  THEN
    Add  $X_{i+1}$  to the tuple
```

Due o più entry sono quindi raggruppate in una tupla ed associate ad uno stesso failure a condizione che la loro distanza temporale sia inferiore della durata della coalescence window. Questa tecnica richiede quindi la scelta della durata della finestra di coalescenza, scelta che viene adoperata mediante un'analisi di sensitività.

L'idea è quella di studiare, al variare della durata della finestra, il numero di tuple generate. A partire da questa informazione si sceglie un durata tale da non causare eccessivi errori di tipo collapse o truncation. Per far ciò si segue una regola empirica che prevede di scegliere una dimensione della finestra in prossimità del "gomito" della curva.

La sensitivity analysis, effettuata tramite uno script matlab, ha prodotto il seguente risultato:

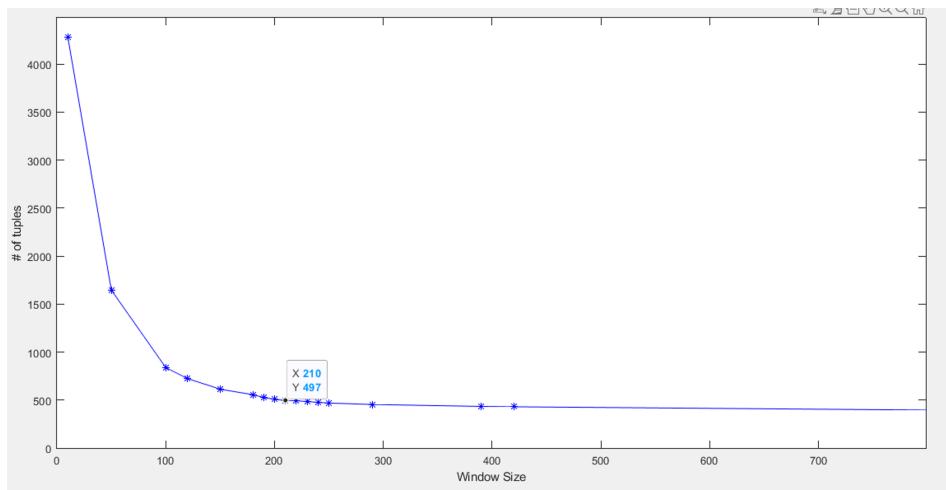


Figura 6.1: Andamento del numero di tuple al variare della durata della finestra di coalescenza.

In accordo alla regola precedentemente descritta abbiamo quindi selezionato la durata della finestra di coalescenza pari a 210s.

Il passo successivo dell’analisi richiede quindi, scelta la durata della finestra di coalescenza, di raggruppare le entry in tuple, ottenendo una distribuzione dei tempi di fallimento dalla quale è possibile ricavare la funzione di reliability empirica del sistema.

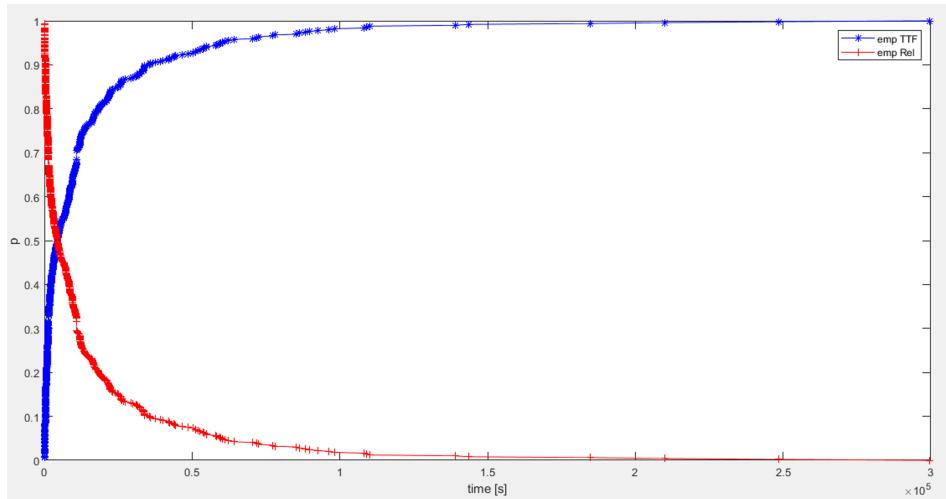


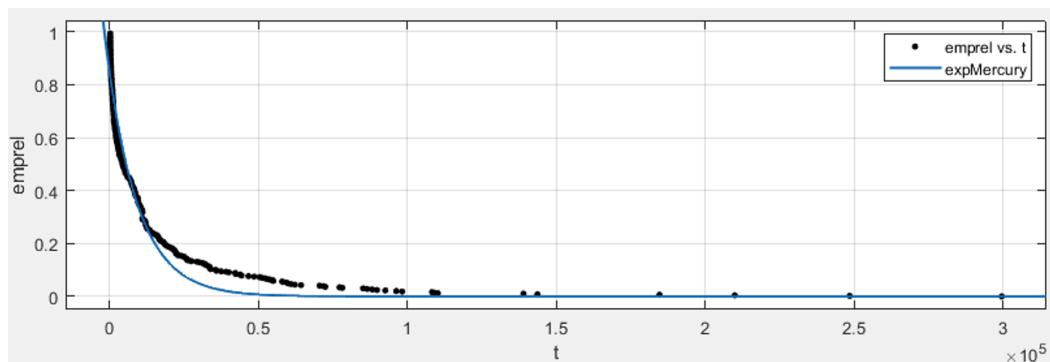
Figura 6.2: Funzioni di reliability ed unreliability empiriche ottenute dall’analisi di Mercury

Ottenuta una funzione di reliability empirica è fondamentale a questo punto analizzarla e verificare la sua somiglianza con delle distribuzioni teoriche note. Nel caso dell'analisi di reliability hanno estrema importanza alcune distribuzioni specifiche:

- Distribuzione esponenziale: associata, tipicamente, a guasti di natura isolata, tipicamente di moduli software o hardware indipendenti. La reliability assume in alcuni casi la struttura di una distribuzione iper-esponenziale, somma di più termini esponenziali, quando i fallimenti sono causati da più moduli distinti ed indipendenti.
- Distribuzione di weibull: associata tipicamente ad errori di accumulazione e failure associati a sistemi in esecuzione per molto tempo. La distribuzione è, infatti, a differenza di quella esponenziale, una distribuzione con memoria.
- Distribuzione lognormale: distribuzione tipica dei fallimenti associati a fault di natura software. Questo tipo di distribuzione è quindi molto importante, poiché in molti sistemi complessi, i fault di tipo software sono quelli più comuni.

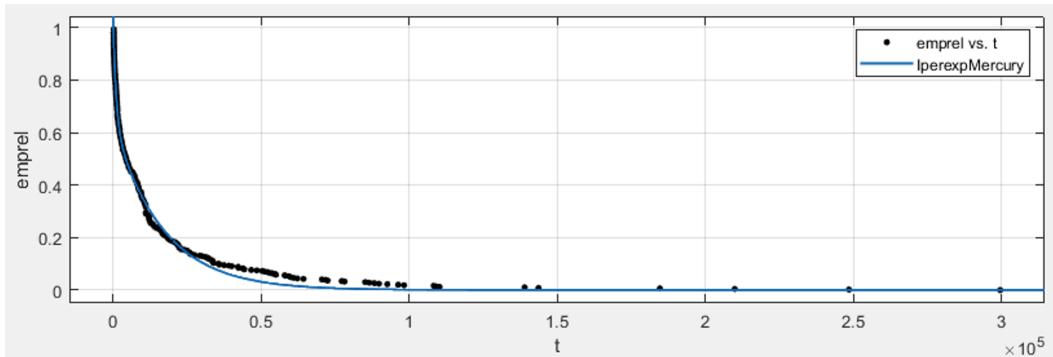
Analizzando la distribuzione empirica ottenuta possiamo quindi ricavare importanti informazioni sulla tipologia di failure che si verificano nel sistema.

Abbiamo quindi inizialmente effettuato il fitting della funzione di reliability con una curva esponenziale:



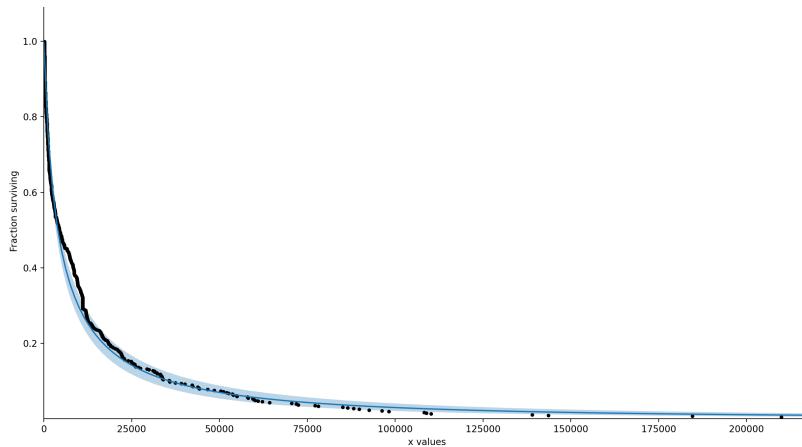
Il risultato, come è già osservabile visivamente non è soddisfacente. Il valore del parametro R^2 è pari a 0.94, ed indica quindi una buona somiglianza con la curva teorica, ma il test statistico di Kolmogorov-Smirnov, utilizzato per verificare la provenienza dei dati dalla distribuzione esponenziale, ha evidenziato un p value estremamente basso, pari a $5.2 * 10^{-5}$, che ci ha portato a rifiutare l'ipotesi nulla del test, ovvero in questo caso l'appartenenza dei campioni ad una popolazione distribuita esponenzialmente.

Abbiamo successivamente tentato il fitting con una funzione iperesponenziale.



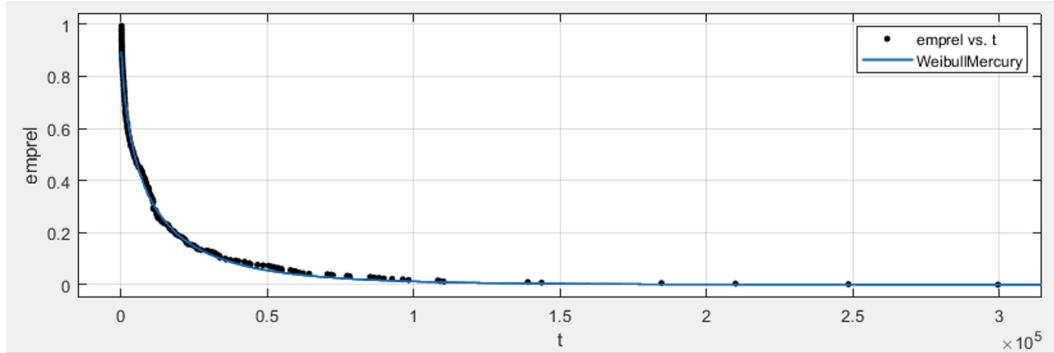
In questo caso il risultato è molto migliore, non solo in termini di R^2 , pari a 0.99, ma soprattutto poichè il test di Kolmogorov-Smirnov restituisce un p-value elevato, pari a 0.7130. In questo caso possiamo quindi concludere che la distribuzione ottenuta è molto vicina ad una iperesponenziale.

Abbiamo poi testato il fitting con una funzione lognormale:



La distribuzione risulta differente da una distribuzione lognormale, il test statistico utilizzato per verificare la natura della distribuzione infatti ci porta a rifiutare l'ipotesi nulla.

L'ultima distribuzione rispetto alla quale siamo andati ad effettuare l'analisi è la weibull distribution:



Analogamente a quanto visto con la distribuzione iper-esponenziale, quella di weibull risulta compatibile con le osservazioni empiriche ottenute, sia in termini di fattore R^2 , estremamente elevato e pari a 0.9917, sia a seguito del test di Kolmogorov-Smirnov, che restituendo un valore pari a 0.0816, non ci permette di rifiutare l'ipotesi nulla.

Distribuzione	R^2	P-Value	H
Esponenziale	0.94	$5.2 * 10^{-5}$	0
Ipersponenziale	0.98	0.71	1
Lognormale ¹	-	-	0
Weibull	0.9917	0.0816	1

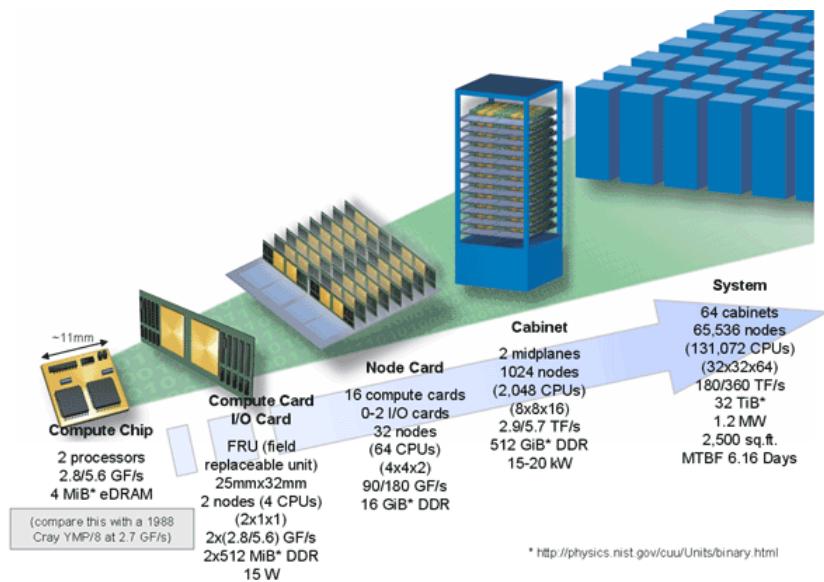
Tabella 6.1: Dati raccolti per le varie distribuzioni.

L'analisi ci permette quindi di giungere alla conclusione che la distribuzione dei fallimenti rilevata tramite l'analisi dei log del supercalcolatore Mercury è molto simile, anche visivamente, ad una distribuzione weibull, ed è quindi

associabile a fault sia di natura hardware che software e ad effetti di accumulazione.

6.2 Analisi dei log di Blue Gene

Blue Gene è un'architettura progettata per realizzare la nuova generazione di supercomputer a parallelismo elevato sviluppati per lavorare con potenze di calcolo che vanno dalle decine di teraFLOPS per arrivare fino al petaFLOPS.



Anche in questo caso il log da analizzare è stato pre-elaborato, mantenendo esclusivamente entry relative a fallimenti fatali, e non è quindi necessaria un'operazione di filtraggio. La metodologia utilizzata è quindi analoga a quanto detto in precedenza per il log associato a Mercury. Il log in questione conta 125624 entry ognuna delle quali ha il seguente formato:

- Timestamp
- Originating node
- Originating card
- Text-free message

La prima fase è quindi anche in questo caso la sensitivity analysis allo scopo di ricavare un valore per la coalescence window. Utilizzando lo stesso script utilizzato in precedenza abbiamo ottenuto il seguente grafico:

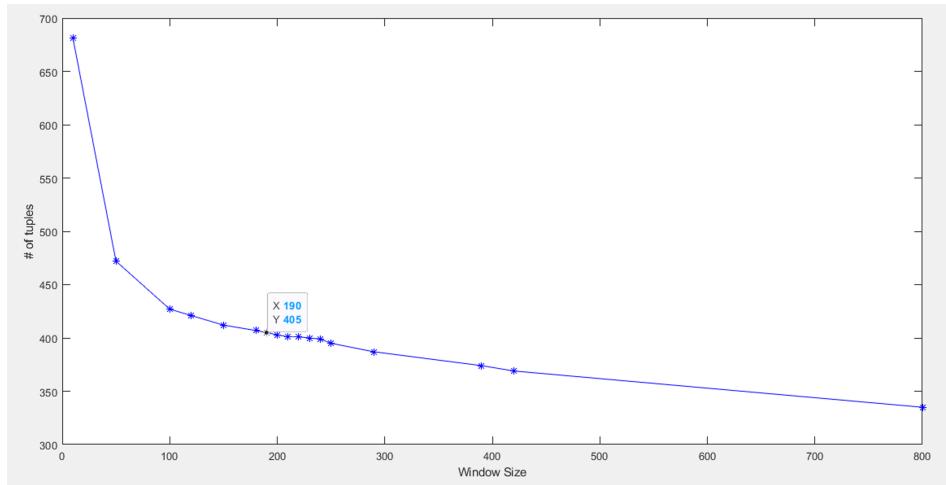


Figura 6.3: Andamento del numero di tuple al variare della durata della finestra di coalescenza.

Seguendo lo stesso principio espresso in precedenza sceglioamo un valore prossimo al "ginocchio" della curva allo scopo di limitare fenomeni di truncation o di collapse. Abbiamo quindi optato di utilizzare una finestra di coalescenza di 190s, valore con il quale otteniamo 405 tuple.

Partendo da questo dato possiamo quindi costruire la distribuzione dei failure e quindi costruire una funzione di reliability empirica. Il risultato ottenuto è il seguente

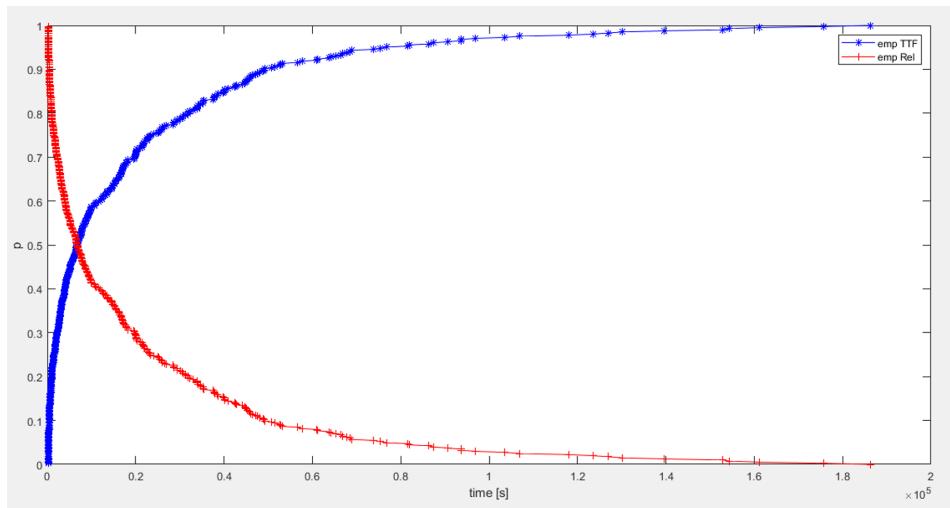
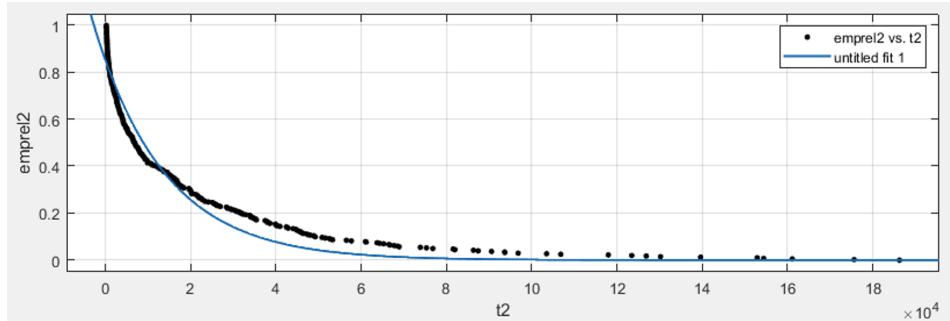


Figura 6.4: Andamento del numero di tuple al variare della durata della finestra di coalescenza.

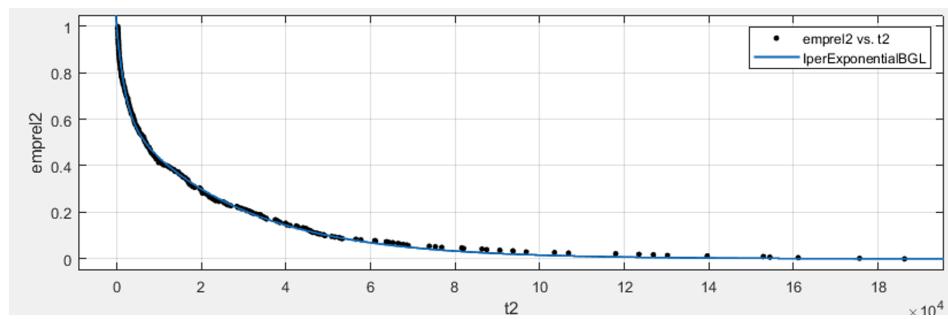
Il passo successivo è anche in questo caso procedere ad un'analisi della distribuzione empirica calcolata cercando di identificare a quale distribuzione teorica corrisponde.

Abbiamo inizialmente tentato il fitting con una funzione esponenziale:



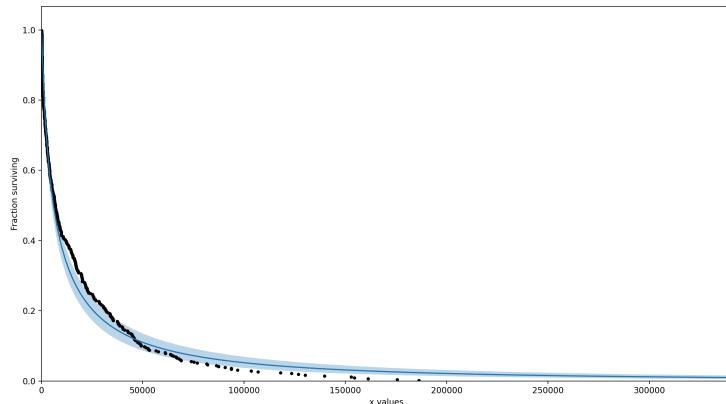
Il risultato in questo caso non è molto positivo, anche visivamente ci accorgiamo che le due distribuzioni hanno alcune differenze. Il valore di R^2 ottenuto è pari a: 0.955. Il test di Kolmogorov-Smirnov restituisce un pvalue basso, pari a: $3.5 * 10^{-4}$, che ci porta quindi a concludere che l'ipotesi nulla di appartenenza dei dati ad una distribuzione esponenziale è rifiutata.

La seconda distribuzione analizzata è quella iperesponenziale:



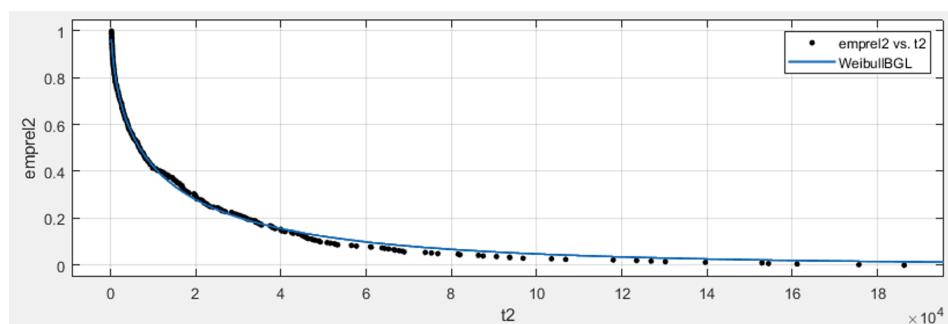
I risultati mostrano, già visivamente, un'elevato grado di similitudine tra le due distribuzioni. Il valore di R^2 è molto elevato e il valore del p-value del test di Kolmogorov-Smirnov, pari a: 0.8971, suggeriscono che la distribuzione è effettivamente molto simile ad una iperesponenziale.

Il fitting con la distribuzione lognormale ha portato al seguente risultato:



Seppur visivamente le due distribuzioni sembrerebbero vicine, il test di ipotesi condotto successivamente porta a rifiutare l'ipotesi nulla che i dati risultino distribuiti come una lognormale.

L'ultima distribuzione analizzata è quella di Weibull:



Visivamente le distribuzioni risultano molto simili. Il valore elevato di R^2 ed il test di Kolmogorov-Smirnov, il cui p-value è pari a: 0.9926, confermano che la distribuzione empirica è simile ad una distribuzione di weibull.

Distribuzione	R^2	P-Value	H
Esponenziale	0.955	$3.5 * 10^{-4}$	0
Ipersponenziale	0.9977	0.8971	1
Lognormale	-	-	0
Weibull	0.993	0.9926	1

Tabella 6.2: Dati raccolti per le varie distribuzioni. DA COMPILARE

I risultati raccolti suggeriscono quindi che la distribuzione dei failure ottenuta analizzando il log è molto simile sia ad una Weibull, suggerendo failure associati a fenomeni di accumulazione e saturazione, sia ad una iper-esponenziale, suggerendo condizioni di failure dovuti a fault di sistemi indipendenti.

6.3 Confronto della reliability dei due sistemi

A partire dalle funzioni di reliability ottenute a partire dall'analisi dei log possiamo a questo punto confrontare l'affidabilità dei due sistemi.

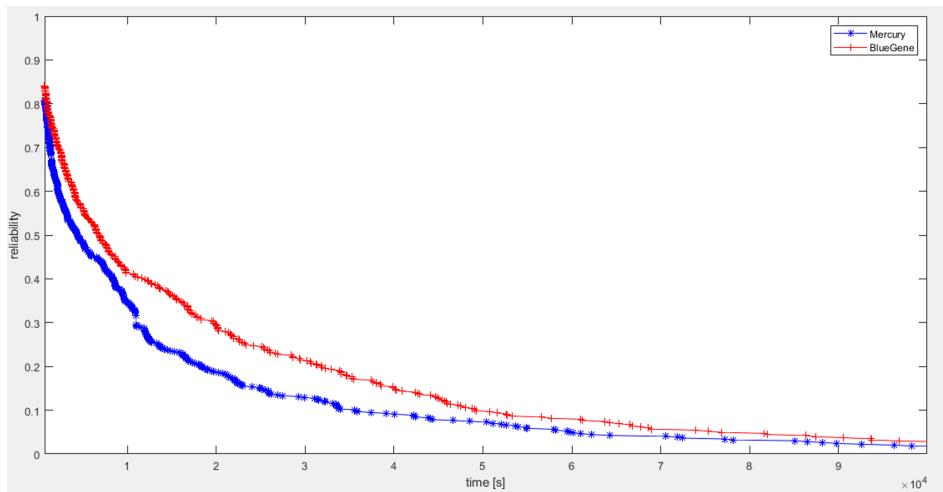


Figura 6.5: Confronto tra le due funzioni di reliability.

Dal grafico notiamo che il sistema BlueGene risulta nel complesso, per ogni mission time, leggermente più affidabile di Mercury.

Abbiamo, inoltre, messo a confronto i $MTTF$ dei due sistemi, ottenuti per integrazione numerica della funzione di reliability. I risultato confermano quanto detto:

- $MTTF_{Mercury} \simeq 1.4 * 10^4 s$
- $MTTF_{BlueGene} \simeq 1.90 * 10^4 s$

6.4 Altre analisi

6.4.1 Scelta della finestra di coalescenza

Lo scopo di questa sezione è cercare di capire se, la scelta operata in termini di durata della finestra di coalescenza per i due log analizzati è generalizzabile a sottosistemi dei due supercalcolatori. Per far ciò l'approccio seguito è il seguente: siamo andati a filtrare i due log forniti per nodo interessato e, nel caso del log di Mercury, anche per tipo di fallimento verificato. Effettuato il filtraggio siamo andati a ripetere l'analisi di sensitività ai log associati ad

i sottosistemi allo scopo di identificare delle dimensioni appropriate per le finestre di coalescenza.

Nel caso di Mercury abbiamo optato per la sensitivity analisys dei seguenti nodi: tg-c401, tg-master, tg-c572, tg-s044, ovvero dei 4 nodi a cui sono associati un maggior numero di entry all'interno del log.

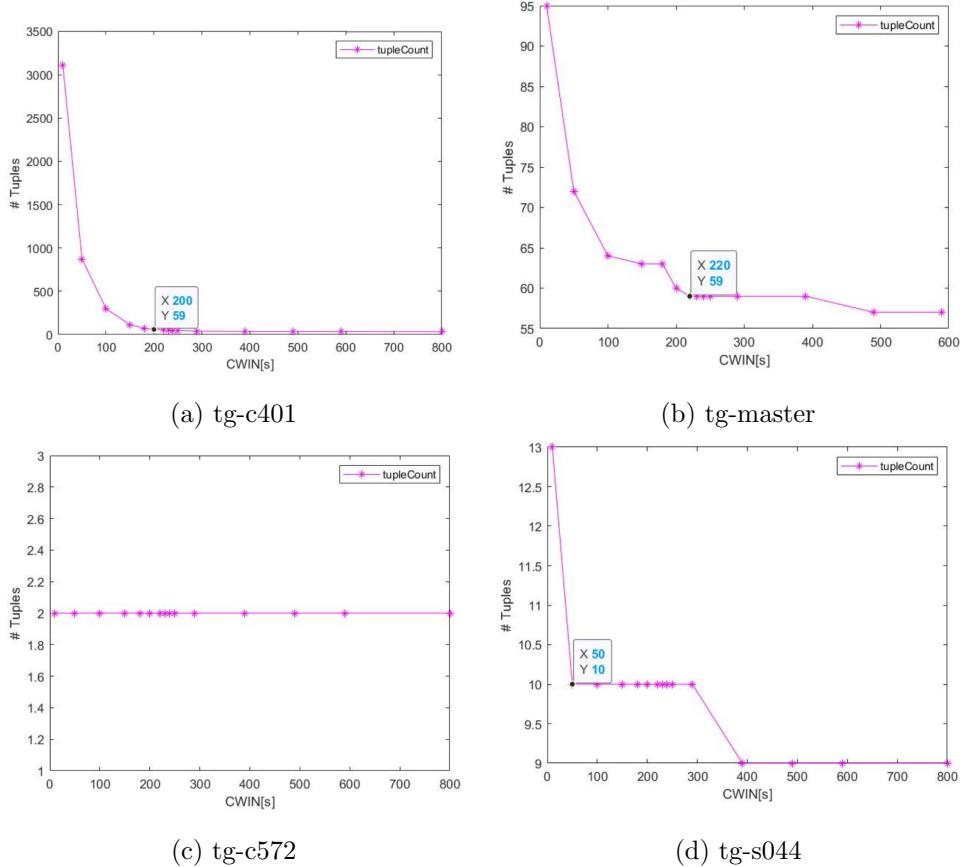
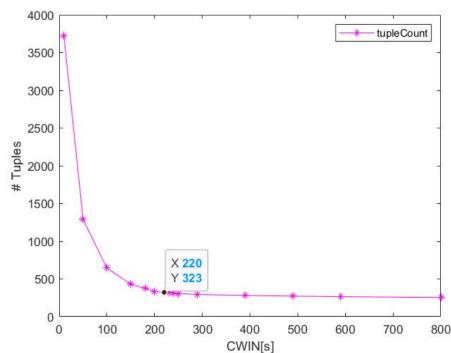


Figura 6.6: Sensitivity analisys per i 4 nodi più frequenti del log di Mercury

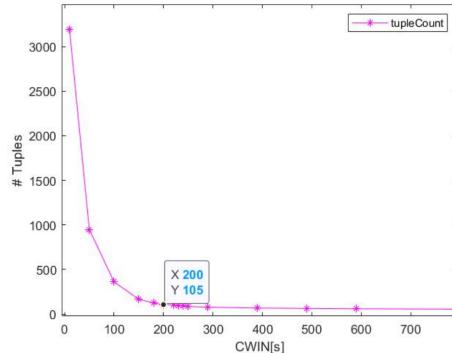
Da questa prima analisi otteniamo un risultato interessante, per i primi due nodi, che come vedremo presentano un numero elevatissimo di entry all'interno del log, una finestra di coalescenza pari circa a $200s$, e quindi prossima a quella scelta per l'analisi del log. Nel caso degli altri due nodi invece il risultato è differente: per il nodo tg-s044 è possibile selezionare una finestra di coalescenza molto più breve, mentre, per il nodo tg-c572 che all'interno del log esclusivamente entry isolate, non ha importanza la durata della finestra scelta. In

generale possiamo concludere quindi che, per i nodi che generano la maggior parte degli eventi registrati all'interno del log, può essere utilizzata una finestra di coalescenza pari a quella utilizzata per l'intero sistema, ma la cosa non si applica necessariamente a sottosistemi che generano una piccola parte delle entry presenti.

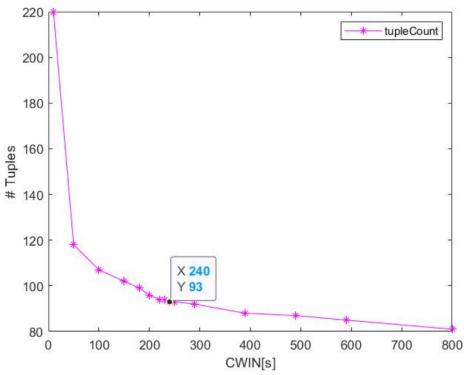
Ripetiamo a questo punto l'analisi operando sulle 6 differenti tipologie di errore registrate all'interno dei log:



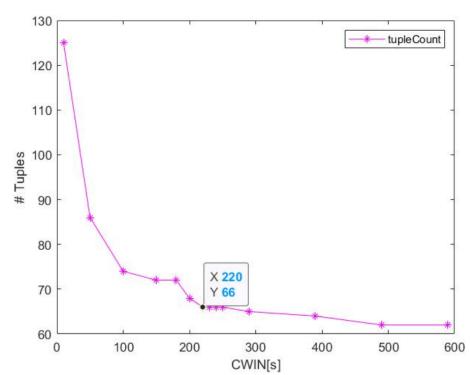
(a) DEV



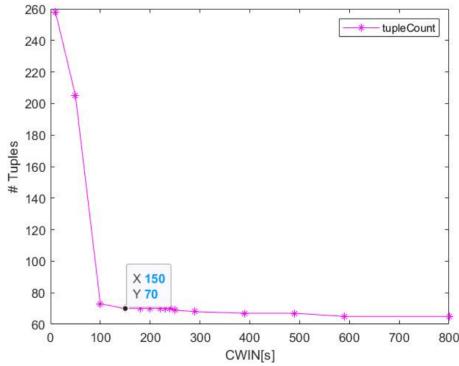
(b) MEM



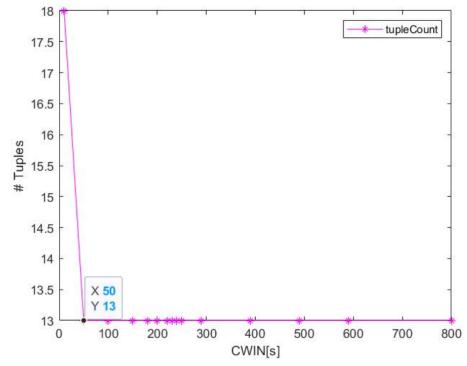
(c) IO



(d) NET



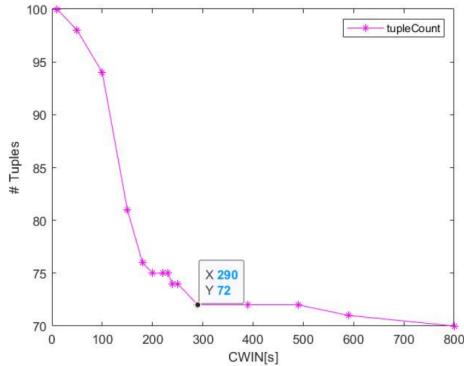
(e) PRO



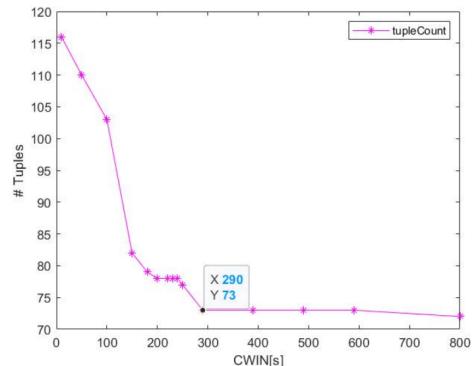
(f) OTH

Il risultato ottenuto è in linea con quanto detto prima: per le classi di errore più frequenti, come DEV e MEM, può essere scelta la stessa durata della finestra di coalescenza del sistema complessivo, prossima a 200s. La cosa non si applica invece al caso delle classi di errore meno frequenti, come OTH o PRO, per le quali è possibile la scelta di una finestra di coalescenza molto più breve.

Abbiamo a questo punto ripetuto l'analisi per i 4 nodi più presenti nel log di Blue Gene, ovvero R71-M0-N4, R12-M0-N0, R63-M0-N2, R03-M1-NF.



(g) R71-M0-N4



(h) R12-M0-N0

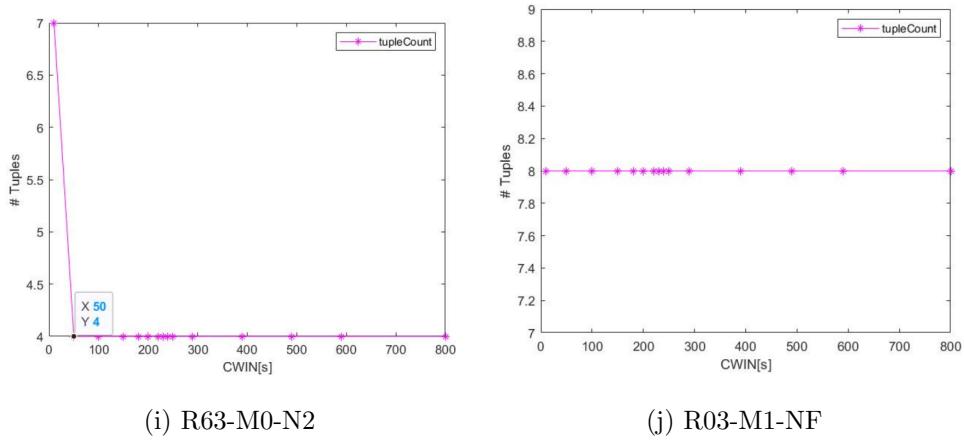


Figura 6.7: Sensitivity analysis per i 4 nodi più frequenti nel log di Blue Gene

I risultati ottenuti mostrano che la scelta utilizzata per la durata della finestra di coalescenza per l'intero sistema non è ben applicabile ai singoli nodi: nel caso di R71-M0-N4 e R12-M0-N0 è opportuno selezionare una durata più lunga, prossima a $290s$, nel caso di R63-M0-N2 è possibile optare per una finestra molto più breve di $50s$, mentre nel caso di R03-M1-NF la durata della finestra non ha influenza poichè questo genera esclusivamente entry isolate.

6.4.2 Reliability bottlenecks

A partire da un'analisi più specifica dei logs è possibile ricercare, all'interno del sistema, i nodi da cui più comunemente sono generati failures.

I 10 nodi che causano più failure all'interno del sistema sono i seguenti:

Nodo	Numero entry log
tg-c401	62340
tg-master	4098
tg-c572	4030
tg-s044	3224
tg-c238	1273
tg-c242	1067
tg-c648	643
tg-login3	382
tg-c117	268
tg-c669	267

Notiamo come, una grandissima percentuale delle entry presenti nel log sono associate ad uno specifico nodo di calcolo, il nodo tg-c401, che si comporta come bottleneck per l'affidabilità del sistema.

Analizzando il log di Mercury è inoltre possibile analizzare le categorie di eventi più comuni:

Categoria	Numero entry log
DEV	57248
MEM	12819
IO	3702
NET	3702
PRO	1504
OTH	34

Notiamo come la maggioranza delle entry presenti nel log siano associate alle categorie DEV e MEM.

Abbiamo ripetuto l'analisi per il log di Blue Gene, i risultati ottenuti sono i seguenti:

Nodo	Numero entry log
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

A differenza di quanto visto per Mercury, analizzando il log di Blue Gene non si identifica un nodo responsabile di molte più entry all'interno del logfile, non sono quindi presenti nodi che fanno da bottleneck per l'esecuzione del sistema.

6.4.3 Correlazione tra nodi e tipi di errore

Come detto in precedenza, ad ogni entry presente nel log di Mercury, è associata la tipologia di errore a cui quella fa riferimento. Sfruttando questa informazione siamo andati a studiare, per i 10 nodi più presenti all'interno del log, la percentuale di errore di ogni tipologia.

Nodo	DEV	MEM	IO	NET	PRO	OTH
tg-c401	50782	11558	0	0	0	0
tg-master	0	0	452	3639	0	0
tg-c572	3176	845	0	0	0	0
tg-s044	0	0	3208	2	0	14
tg-c238	1071	0	230	3	0	0
tg-c242	918	149	0	0	0	0
tg-c648	643	0	0	0	616	0
tg-login3	0	0	381	1	0	0
tg-c117	263	5	0	0	0	0
tg-c669	257	10	0	0	0	0

A partire da questi dati è stata costruita la percentuale di ogni tipologia di errori per ogni categoria di nodo:

Nodo	DEV	MEM	IO	NET	PRO	OTH
Calcolo	80%	18.2%	0	0	1.8%	0
Master	0	0	12%	88%	0	0
Storage	0	0	99%	0.1%	0	0.9%
Login	0	0	99.7%	0.3%	0	0

Ottenuti questi dati risultano evidenti alcune proprietà del sistema.

- I nodi di calcolo sono tipicamente soggetti ad errori di tipo DEV, molto frequenti, o MEM.
- Il nodo master è invece maggiormente soggetto ad errori di tipo IO o NET. Questo comportamento è probabilmente spiegato dalla necessità di questo nodo di comunicare molto con gli altri per orchestrare il comportamento.

- I nodi di tipo login e di storage sono maggiormente soggetti ad errori di IO. Questo comportamento è spiegato dalle comuni interazioni che questa tipologia di nodi effettua con dispositivi esterni come hard-disk.

Notiamo inoltre alcune particolarità nel nodo tg-c648, unico nodo che presenta un elevato numero di errori di tipo PRO.

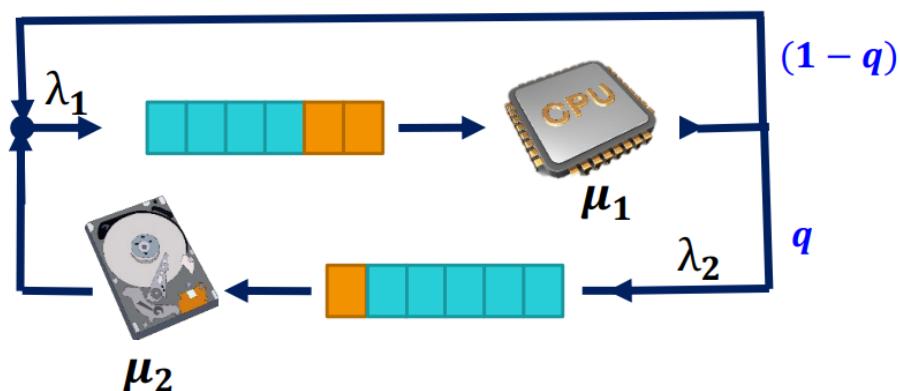
Capitolo 7

Teoria delle code

7.1 Sistema in analisi

Lo scopo di questo esercizio è l'utilizzo di tecniche model-based analitiche basate sulla teoria delle code allo scopo di studiare le performance di un sistema di elaborazione composto da una CPU e da un modulo di IO. Del sistema vogliamo in particolar modo studiare, al variare del numero di processi in esecuzione, i tassi di arrivo per ogni coda, il numero medio dei processi in attesa ed il tempo di risposta medio.

Il sistema sotto analisi è il seguente:



Dato	Valore
q	$\frac{1}{3}$
μ_{CPU}	5hz
μ_{IO}	2hz

Si tratta di una rete di code chiusa, l'approccio utilizzato per studiarla è quindi quello della mean value analysis. Questa tecnica consente, utilizzando la proprietà dell'osservatore esterno, di definire una relazione ricorsiva che consente, note le proprietà del sistema con K processi in esecuzione di calcolare lo stato del sistema con $K+1$ processi in esecuzione. Il caso base per $K = 0$ processi è inoltre banalmente risolto.

7.2 Metodologia e risultati ottenuti

L'algoritmo utilizzato per l'analisi è quindi il seguente:

Caso base:

$$\overline{N}_i[0] = 0$$

Passo ricorsivo:

$$\begin{aligned} \overline{T}_i[K] &= (1 + \overline{N}_i[K - 1]) \frac{1}{\mu_i} \\ \overline{N}_i[K] &= K \frac{\hat{\lambda}_i \overline{T}_i[K]}{\sum_{j=1}^M \hat{\lambda}_j \overline{T}_j[K]} \\ \lambda_i[K] &= \frac{\overline{N}_i[K]}{\overline{T}_i[K]} \end{aligned}$$

I parametri $\hat{\lambda}_i$ per la rete sotto analisi possono essere facilmente ricavati dalla relazione $\lambda_2 = q\lambda_1$.

A partire da questo algoritmo ricorsivo, implementato in python, abbiamo quindi ottenuto i seguenti risultati:

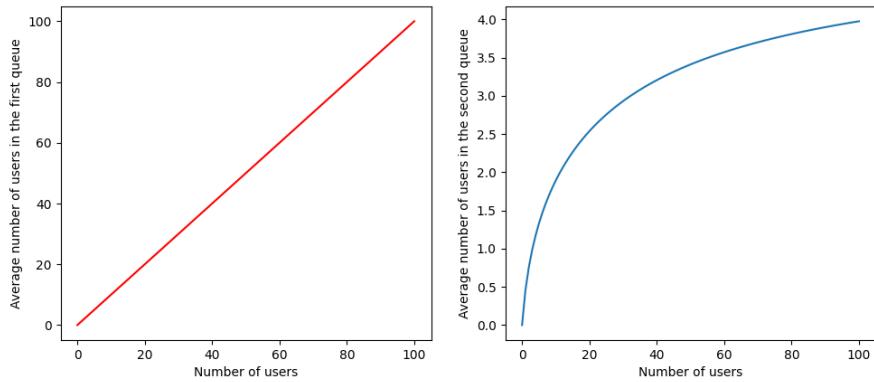


Figura 7.1: A sinistra il numero medio di processi in coda per il processore, a destra per l'IO.

Il numero medio di utenti presenti nelle due code cresce al crescere del numero di utenti. Nonostante il processore sia nel complesso più veloce, la frazione di processi che deve servire è più elevata. Ne segue che il numero di processi in coda per il processore è molto più elevato e cresce più velocemente.

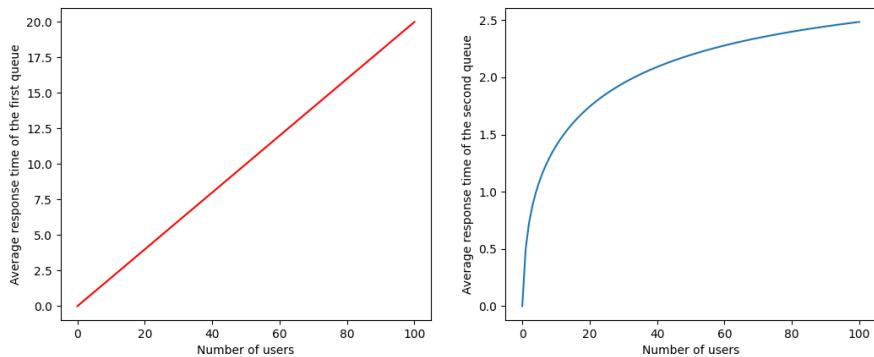
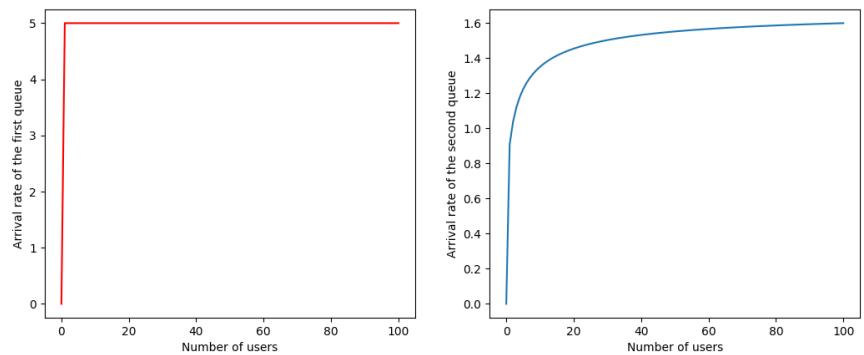


Figura 7.2: A sinistra il tempo medio di risposta per il processore, a destra per l'IO.

L'andamento del tempo di risposta medio è sostanzialmente analogo, ed è maggiore in generale per il processore che per l'IO per le stesse motivazioni del numero medio di utenti.



L'arrival rate del processore si assesta rapidamente al valore 5, mentre cresce all'aumentare del numero di utenti nel caso del sistema di IO.