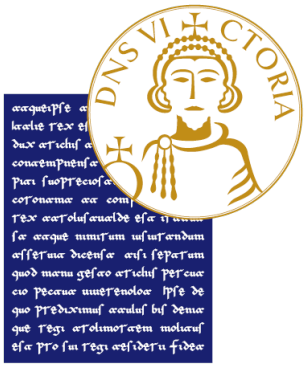


Università degli Studi del Sannio

Dipartimento di Ingegneria

Corso di Laurea Magistrale in Ingegneria Informatica

Data science

Homework 1

Prof:
Pecchia Antonio

Studenti:
Cinelli Jessica, 399000529
Mazzitelli Francesco C., 399000532

Anno Accademico 2022-2023

Indice

Elenco delle figure

1 Introduzione

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. Come tutti i sistemi di calcolo parallelo ad alte prestazioni, raggruppati in cluster, anche l'architettura Blue Gene non è esente da problemi.
Lo scopo di questo homework è quindi quello di riuscire a costruire un reliability model dell'architettura in modo da riuscire a prevedere statisticamente dopo quanto tempo si verifica un fault assumendo di trovarsi in uno stato di corretto funzionamento del sistema.
Gli strumenti e i dati utilizzati per poter effettuare questo studio sono:

- **Dataset** contenente i log di due mensilità di failures del sistema
- **R Studio** che ha rappresentato il supporto di elaborazione principale
- **Script Python**:
  - **cwinAnalysis.py**: script che valuta la sensibilità del conteggio delle tuple rispetto alla finestra di coalescenza. Analizza quindi come varia il conteggio delle tuple incrementando il valore di finestratura
  - **logCoalescence.py**: script utilizzato per poter effettuare la coalescenza del log, raggruppando le ridondanze i tuple, in relazione alla finestratura effettuata precedentemente
  - **statistics.py**: script che genera l'elenco dei primi 20 nodi in relazione al numero di failures ad essi associati

2 Passo 0

2.1 Scelta dei file

Il primo passo svolto per la realizzazione del seguente homework è stato quello di identificare la coppia di file da utilizzare.
Il criterio di selezione ha previsto un semplice calcolo basato sui numeri finali delle matricole degli studenti facenti parte del gruppo.
Questi numeri sono stati quindi sommati e, come da istruzioni, ne è stato calcolato il modulo in relazione al numero di partecipanti, ottenendo così il valore rappresentante la coppia di file associati al gruppo.

Nello specifico le matricole utilizzate per il calcolo sono state:

- 399000532
- 399000529

Effettuando i calcoli:

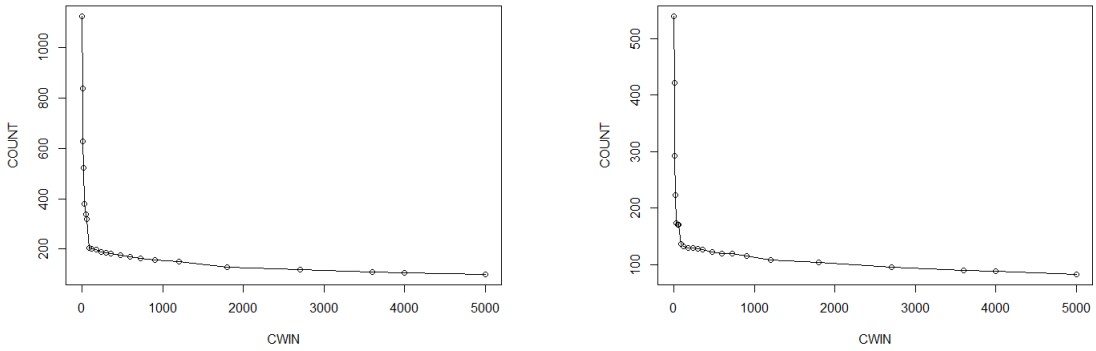
- **Somma**: $2 + 9 = 11$
- **Modulo**: $11 \ \% \ 2 = 1$

La coppia di file selezionati è stata quindi quella marchiata con il valore 1, ovvero la coppia relativa alle mensilità di **Settembre** e **Ottobre**.

3 Studio della reliability

3.1 Creazione delle Tuple

Da una prima analisi dei file è stato possibile constatare che gran parte degli eventi registrati nel file di log avessero lo stesso timestamp dei successivi. Appurato il fatto che ci fosse una forte correlazione tra gli eventi, è stato ritenuto opportuno analizzare simultaneamente tutti i failure che avvenissero in un determinato intervallo temporale. L'intervallo, o finestratura, è stato scelto tramite un vero e proprio processo di analisi. La prima operazione svolta è stata l'applicazione ai file di interesse dello script python chiamato cwinAnalysis.py. Questo script restituisce in output un file contente delle finestre di varie dimensioni a cui sono associati i conteggi dei failure. Dopo la generazione del file di output, i valori sono stati inizialmente studiati in RStudio effettuandone lo scatterplot (Appendice A).

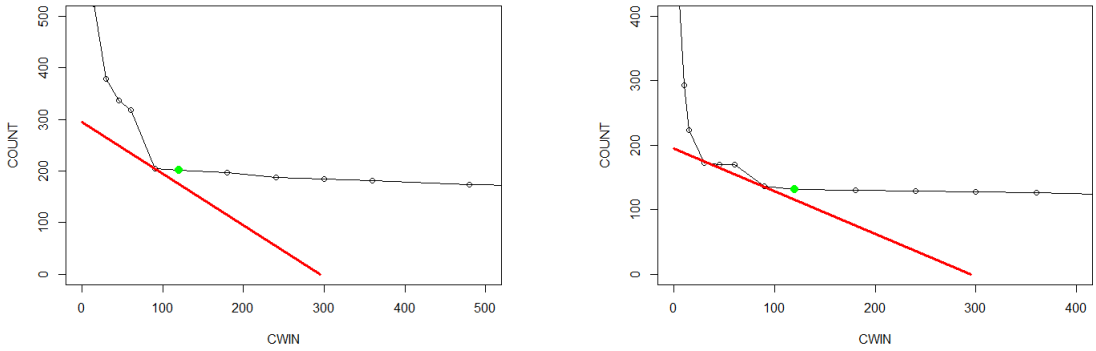


((a)) Settembre

((b)) Ottobre

Figura 1: Scatterplot del numero di malfunzionamenti in funzione della finestra

Come si evince dalla figura 1 , all'aumentare della dimensione della finestra tende a diminuire il numero di tuple, le quali rappresentano una stima del numero di eventi di malfunzionamento in un determinato arco temporale. Per selezionare il punto rappresentante la finestratura migliore è stato il punto della curva in cui l'andamento tende ad essere stazionario, ovvero è stato selezionato un valore di CWIN nell'intorno destro alla tangente passante per il punto di ginocchio della distribuzione. L'euristica utilizzata mira a trovare un valore situato nella regione della curva che presenta un andamento relativamente costante, con l'obiettivo di garantire la stabilità del sistema. Allo stesso tempo, questo valore deve essere abbastanza piccolo da prevenire le collisioni tra i componenti del sistema.



((a)) Settembre

((b)) Ottobre

Figura 2: Selezione del punto di finestratura ottimale

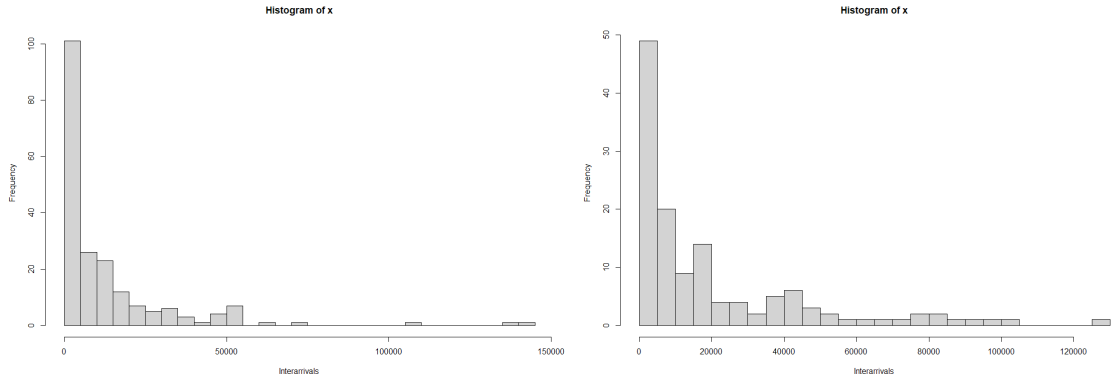
Dopo aver individuato le finestre di coalescenza (figura 2), tramite lo script python logCoalescence.py sono state generate:

- Le tuple in cui sono stati raggruppati gli eventi verificatisi nell'intervallo selezionato;
- Gli interarrivi in cui sono stati riportati gli intervalli temporali che intercorrono tra le una tupla e la successiva.

3.2 Analisi univariata degli interarrivi

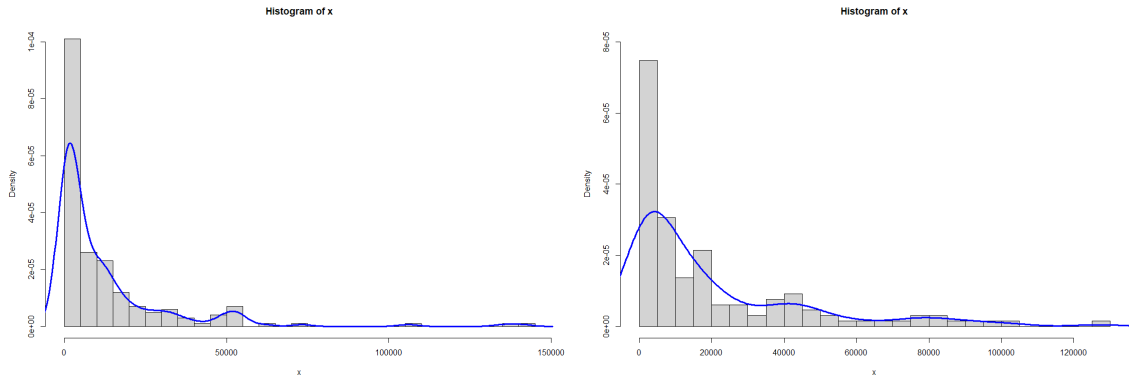
Lo script logCoalescence.py oltre a generare le tuple, genera un file chiamato interarrivals.txt contenente gli interarrivi, ovvero la differenza tra il timestamp della finestra corrente e il timestamp della finestra successiva. Il file i*nterarrivals* è il dato trasformato, cioè il dato su cui possiamo applicare algoritmi di data mining.
Per riuscire ad estrarre informazioni è stato ritenuto opportuno condurre un'analisi univariata, in modo da poter individuare nuovi valori quali: media, deviazione standard, calcolo range semi-inter-quartile e intervalli di confidenza. I risultati sono riportati nelle tabelle 1 e 2.

Per condurre l'analisi è stato prodotto lo script R riportato in Appendice B, applicabile indistintamente ad entrambi i file. Successivamente a questo tipo di analisi si è passati a soluzioni di tipo visuale.



((a)) Settembre ((b)) Ottobre

Figura 3: Istogramma-Frequenze

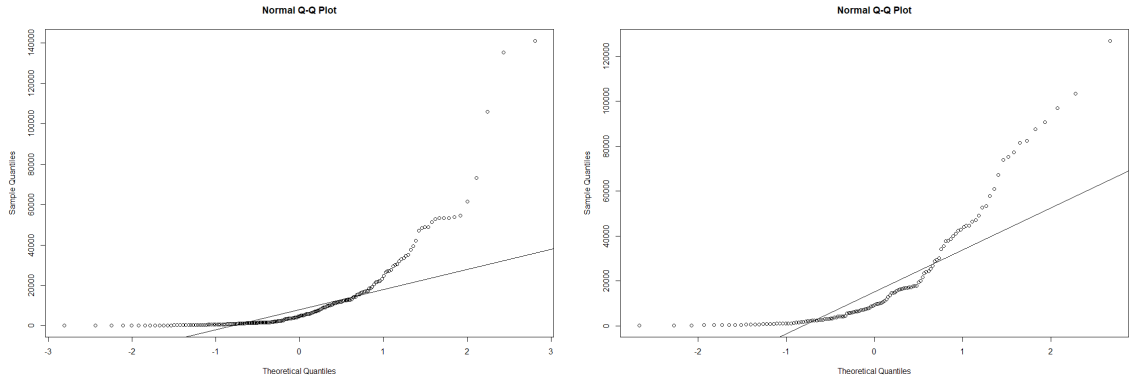


((a)) Settembre ((b)) Ottobre

Figura 4: Istogramma-Densità

Già da una prima analisi degli istogrammi in figura 3 e 4 si nota come gli intervalli si distribuiscano con una distribuzione fortemente skewed: la maggior parte delle osservazioni si colloca negli intervalli a sinistra. Gli intervalli alla fine del range osservato sono molto poco popolati.

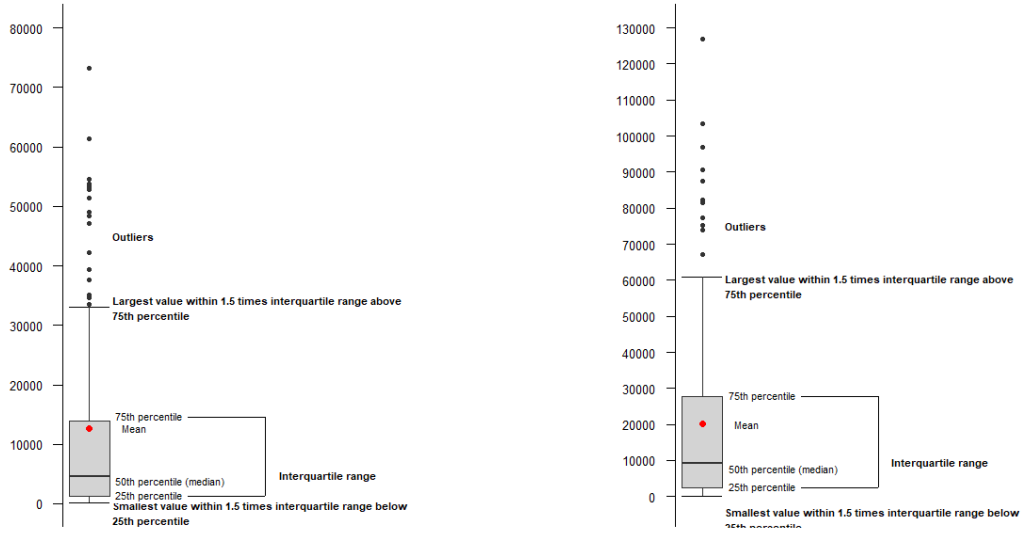
Per capire se i dati sono distribuiti come una gaussiana è stato effettuato il plot Quantile-Quantile.



((a)) Settembre ((b)) Ottobre

Figura 5: Q-Q Plot

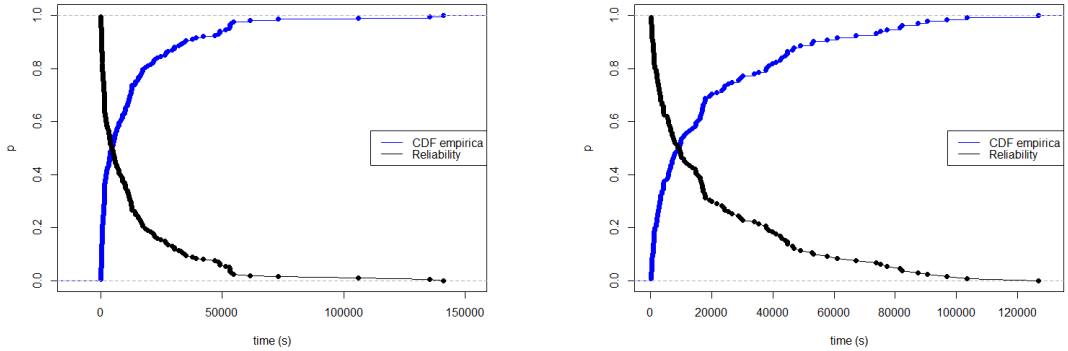
Come possiamo notare dalla figura 5, in entrambi i casi i dati non provengono da una distribuzione normale; infatti, la maggior parte delle coppie *(quantile teorico, quantile osservato)* non si trova in corrispondenza della retta.



((a)) Settembre ((b)) Ottobre

Figura 6: Boxplot

La figura 6, mostra l'evoluzione della distribuzione delimitata dal primo e terzo quartile, al centro è possibile notare la mediana e alle due estremità i baffi, rappresentanti il valore minimo e il massimo relativi alla distribuzione delimitata. In questo modo è possibile individuare agevolemente gli outliers, la cui presenza risulta essere abbondante in entrambe le mensilità osservate. In particolare, osserviamo che la distribuzione presenta molti outlier, di conseguenza la media ha un valore poco rappresentativo ai fini dell'analisi.



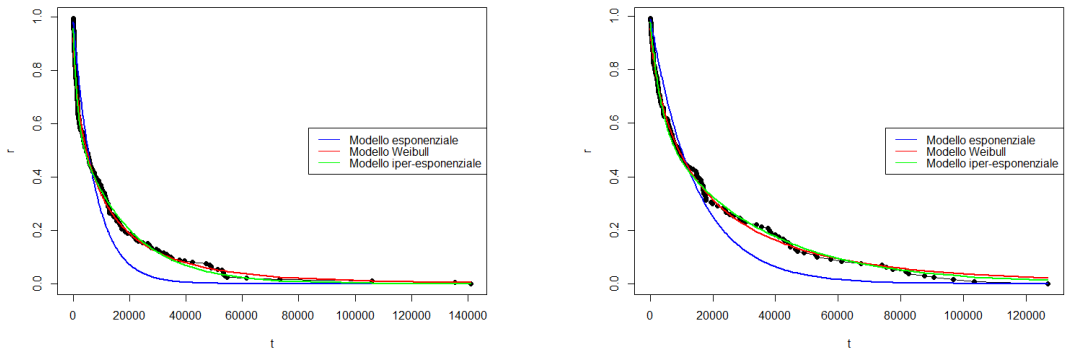
((a)) Settembre ((b)) Ottobre

Figura 7: CDF empirica e reliability

La figura 7 riporta per ogni mese sia la CDF delle osservazioni dei TTF calcolati in precedenza sia la reliability vista come il complemento della prima funzione. In entrambi i mesi la reliability (curva nera) decresce all'aumentare del tempo in conseguenza al fatto che la probabilità che si sia verificato un failure (curva blu) aumenta al crescere dell'intervallo temporale.

3.3 Modelli di Reliability

A partire dai valori degli interarrivi è stata calcolata la ecdf per individuare la probabilità che si verifichi un failure. Siccome si tratta di un valore probabilistico e l'obiettivo è quello di costruire un modello di reliability, è possibile, a partire dalla ecdf, calcolare la probabilità che non si verifichi un evento di failure. Per studiare con quale andamento decresce la funzione della reliability, sono stati prodotti ed analizzati i modelli regressivi non lineari: il modello esponenziale, il modello di Weibull e il modello iper-esponenziale. Per ogni modello è stato effettuato il test di Kolmogorov-Smirnov. Per condurre l'analisi è stato prodotto lo script R riportato in Appendice C,



((a)) Settembre

((b)) Ottobre

Figura 8: Modelli di regressione non lineare

Dalla figura 8 si evince che :

- Il modello regressivo esponenziale relativo al mese di settembre non risulta essere valido, infatti il test di Kolmogorov-Smirnov ha prodotto un valore del p-value inferiore allo 0.05;
- Il modello regressivo basato sulla distribuzione di Weibull e il modello iper-esponenziale, per entrambi i mesi, si adattano meglio alle osservazioni sperimentali raccolte. In particolare:
  - Il modello iper-eponenziale approssima al meglio la reliability del mese di settembre. Il test di Kolmogorov-Smirnov ha prodotto un p-value di 0.9997.
  - Il modello Weibull e il modello iper-esponenziale approssimano entrambi al meglio la reliability del mese di ottobre. Per i due modelli, il test di Kolmogorov-Smirnov ha prodotto un p-value identico pari a 0.9991.

3.4 Risultati ottenuti

I risultati delle elaborazioni descritte nelle sezioni precenti sono stati riassunti nelle tabelle 1 e 2.

Coalescence window	120	Tuple count	201
MTTF	Mean: 12667.81 Sd: 20501.46 Median: 4676.5 SIQR: 6706	MTTF (90% Conf) MTTF (95% Conf)	[10272.16, 15063.46] [9809.119, 15526.5]
Exponential model parameters	λ: 7.894024e-05	KS Test	p-value: 0.005657 reject: yes
Weibull model parmeters	λ: 7.894024e-05 α: 0.95	KS Test	p-value: 0.7024 reject: no
Hyperexponential model parameters	α_1: 0.4 λ_1: 7.282e-05 α_2: 0.6 λ_2: 7.282e-06	KS Test	p-value: 0.9997 reject: no BEST FIT

Tabella 1: Failures di Settembre

Coalescence window	120	Tuple count	132
MTTF	Mean: 20144.95 Sd: 25742.38 Median: 9296 SIQR: 12608.5	MTTF (90% Conf) MTTF (95% Conf)	[16418.92, 23870.98] [15695.33, 24594.57]
Exponential model parameters	λ_1: 4.964024e-05	KS Test	p-value: 0.1685 reject: no
Weibull model parmeters	λ: 4.964024e-05 α: 0.95	KS Test	p-value: 0.9991 reject: no BEST FIT
Hyperexponential model parameters	α_1: 0.4 λ_1: 4.964024e-05 α_2: 0.6 λ_2: 4.964024e-06	KS Test	p-value: 0.9991 reject: no BEST FIT

Tabella 2: Failures di Ottobre

4 Itemset frequenti

4.1 Creazione Transazioni

Altra operazione svolta è stata l'applicazione dell'algoritmo "Apriori" per il riconoscimento degli itemset frequenti. Lo scopo di questa analisi è stato quello di identificare gli insiemi di nodi del sistema "Blue Gene" che falliscono simultaneamente e provare a dare una spiegazione a questo fenomeno ipotizzando una possibile correlazione tra essi.

Per poter svolgere questa tipologia di analisi è stato però necessario raggruppare le tuple ottenute come risultato dalle analisi precedenti, in delle transazioni. Queste corrispondono all'insieme di failures che si verificano nello stesso intervallo di tempo e sono state strutturate in modo tale da non considerare più occorrenze di failure dello stesso nodo all'interno della singola transazione.

L'operazione di creazione è stata effettuata tramite uno script python denominato **transactions.py** la cui implementazione è riportata nell'appendice D.

Tale script ha restituito le transazioni create all'interno di un file denominato "output.csv", salvato all'interno della directory associata alla mensilità d'interesse.

Infine è stato prodotto uno script R relativo all'analisi degli itemset frequenti, la cui implementazione è riportata nell'appendice E. Inizialmente sono state individuate due soglie di supporto secondo il criterio:

$$\frac{numero\_failures\_giornalieri \cdot singola\_mensilità}{totale\_failures}$$

Supporto Settembre	s_sep<-2*30/length(data_sep)
Supporto Ottobre	s_oct<-1*30/length(data_oct)

Successivamente, per via di problemi relativi all'esecuzione di apriori, è stata effettuata un'operazione di fine tuning empirico con l'obiettivo di individuare i valori che più si avvicinassero alla generazione dei risultati, e quindi delle regole associative, attesi. I valori di supporto selezionati sono stati:

Supporto Settembre	0.298
Supporto Ottobre	0.257

Le due soglie di supporto sono state utilizzate per la costruzione delle regole associative generate dalla funzione apriori. I due set di regole sono stati poi visualizzati graficamente.

4.2 Risultati Market Basket Analysis

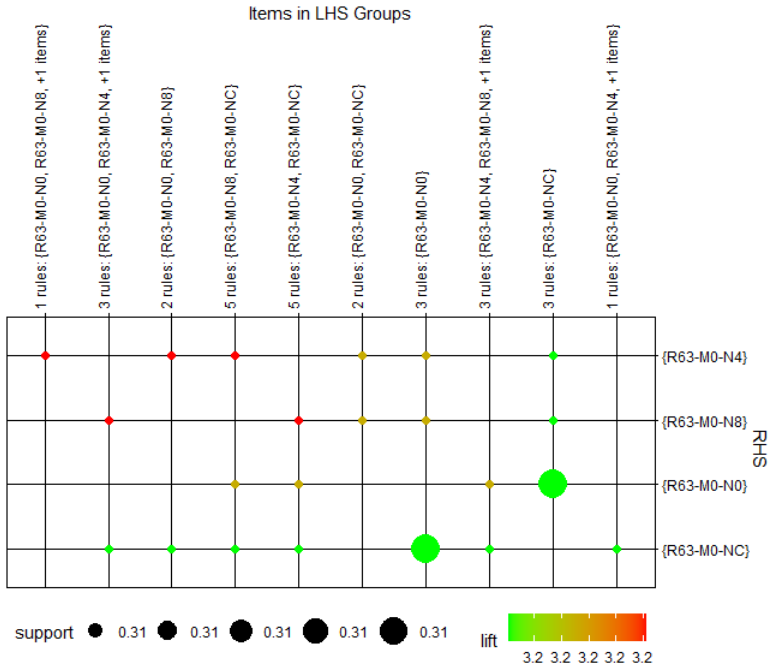
I parametri scelti per l'applicazione dell'algoritmo sono riassunti nella tabella 3.

Tabella 3: Applicazione algoritmo Apriori

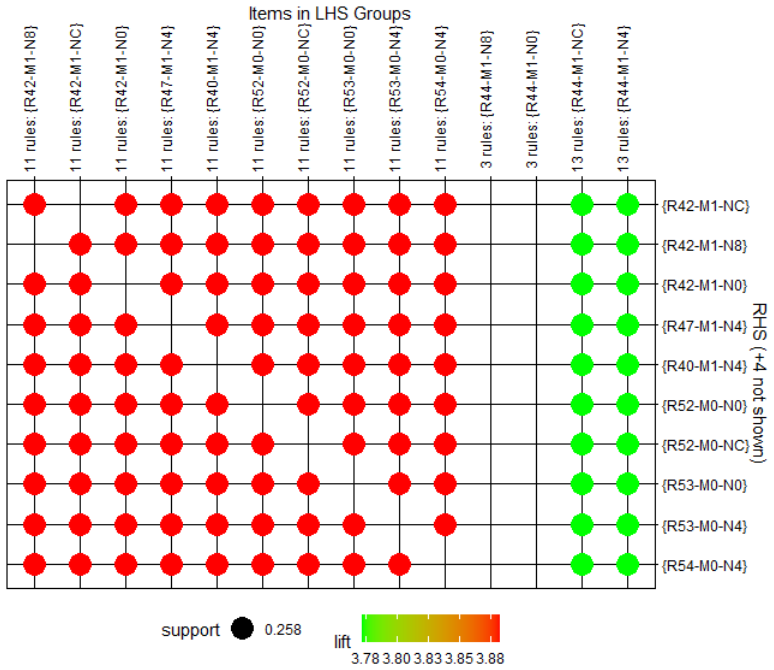
	Supporto	**Confidenza**	**Minlen**	**Maxlen**
Settembre	0.298	0.75	2	20
Ottobre	0.257	0.75	2	20

L'algoritmo ha prodotto un totale di 28 regole e 24590 regole rispettivamente per il mese di Settembre e Ottobre.
Poiché per le transazioni relative ai log del mese di Ottobre l'algoritmo Apriori ha restituito un numero molto elevato di regole, si è deciso di rimuovere le regole ridondanti utilizzando la funzione R *is.redundant(transaction)*. In questo modo il numero di regole si è ridotto a 142.
I risultati sono stati riportati all'interno di un unico grafico rappresentante l'incidenza degli itemset frequenti con la regola associativa ad essi correlata e il lift, un indicatore rappresentante la bontà della regola estratta. Le colonne rappresentano gli items antecedenti della regola associativa e le righe rappresentano gli items conseguenti; il colore dell'intersezione riga-colonna rappresenta la misura del lift e la dimensione mostra il supporto. Dalla figura 9 riportante i risultati è possibile notare che

- in entrambi i mesi le regole presentano un valore di lift molto elevato: ciò significa che gli eventi sono dipendenti tra loro;
- la maggior parte degli itemset frequenti estratti fanno riferimento ad una tipologia di nodi particolare. I nodi N0, N4 ed NC sono caratterizzati delle componenti hardware dedicate alle operazini di I/O, questo lascia supporre che la maggior parte dei failure dell'intero sistema Blue Gene, facciano riferimento ad operazioni di inter-comunicazione tra i vari nodi del sistema, come ad esempio problemi di sincronizzazione della comunicazione.



((a)) Settembre



((b)) Ottobre

Figura 9: Risultati Market Basket Analysis

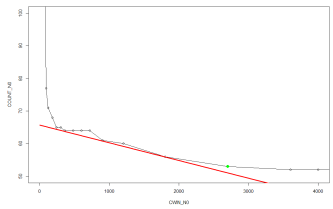
5 Studio della reliability a livello di nodo

5.1 Unione dei file e creazione dei modelli

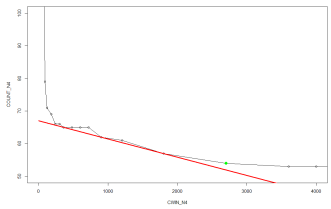
La seconda parte dello studio ha previsto una tipologia di analisi parallela alla prima. La differenza principale consta però nella granularità dell'oggetto di studio, che in questo caso è più fine, in quanto non si analizzeranno i failures relativi all'intero sistema Blue Gene ma solo i nodi che falliscono con frequenza maggiore.

Per realizzare ciò sono stati inizialmente utilizzati i comandi bash: **cat** e **wc** e succesivamente lo script python **statistics.py** responsabile del conteggio delle occorrenze.

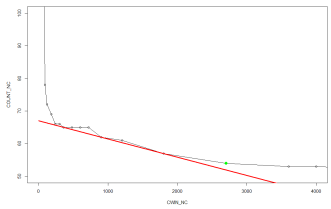
Successivamente è stato creato uno script python denominato **nodeSelection.py**, la cui implementazione è riportata nell'Appendice F, al quale sono stati passati come parametri l'intero file unito e i primi 6 nodi ottenuti come risultato dallo script statistcs.py. Questo ha quindi prodotto in output 6 file contenenti tutti i failures associati al nodo passato come parametro. Una volta individuato ciò si è proceduto con lo stesso procedimento svolto per l'analisi dei modelli di reliability.
I file sono stati sottoposti ad un'operazione di finestratura per riuscire ad individuare i failures avvenuti nello stesso intervallo di tempo. Lo studio della finestratura ha consentito la scelta dell'intervallo ottimale.



((a)) R62-M0-N0

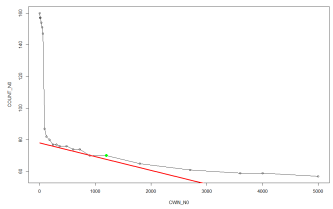


((b)) R62-M0-N4

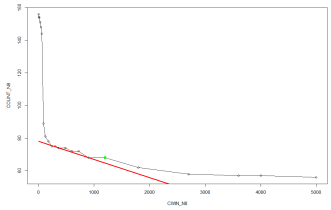


((c)) R62-M0-NC

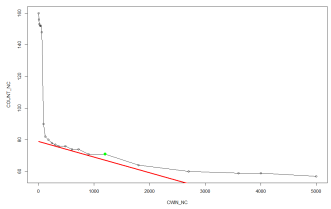
Figura 10: Selezione intervallo ottimale per R62-M0



((a)) R63-M1-N0



((b)) R63-M1-N8



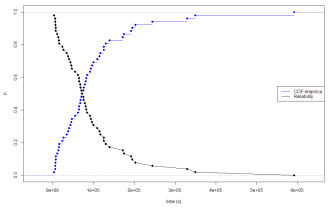
((c)) R63-M1-NC

Figura 11: Selezione intervallo ottimale per R63-M1

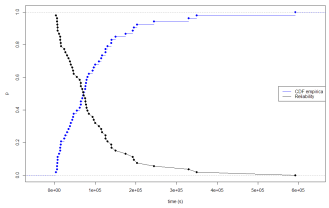
Una caratteristica interessante pervenuta da questo studio è il fatto che **i valori di finestratura ottimale sono gli stessi per tutti i nodi appartenenti allo stesso rack**. Come si può vedere dalle figure 10 e 11 i punti selezionati sono:

Intervallo R62-M0	2700
Intervallo R63-M1	1200

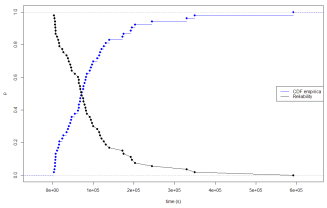
Successivamente si è passati all'operazione di raggruppamento dei failures in tuple, in relazione all'intervallo temporale individuato precedentemente e analisi degli interarrivi. Questo ha quindi portato all'individuazione dell'ECDF indicante la probabilità che si verifichi un failure all'aumentare del tempo:



((a)) R62-M0-N0

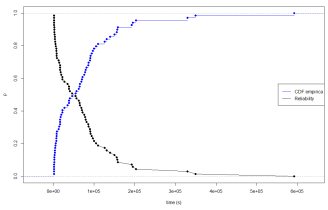


((b)) R62-M0-N4

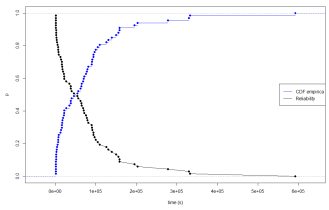


((c)) R62-M0-NC

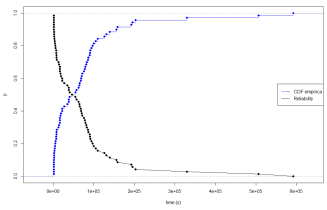
Figura 12: Plot MTTF per R62-M0



((a)) R63-M1-N0



((b)) R63-M1-N8

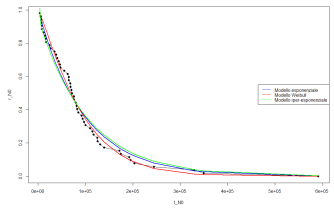


((c)) R63-M1-NC

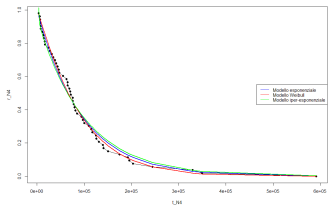
Figura 13: Plot MTTF per R63-M1

Anche in questo caso, come si può notare dalle figure 12 e 13, la forma acquisita dalle ECDF, indicanti il MTTF, risulta essere quasi sovrapponibile.

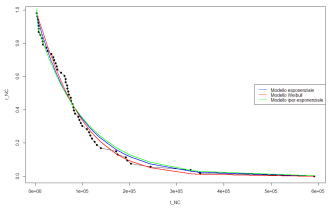
Infine, come ultimo passo di questa analisi, sono stati individuati i modelli di reliability tramite regressioni non lineari.



((a)) R62-M0-N0

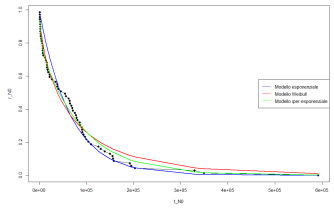


((b)) R62-M0-N4

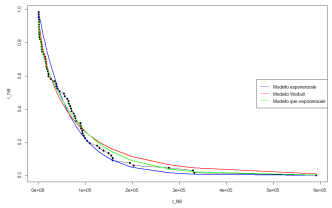


((c)) R62-M0-NC

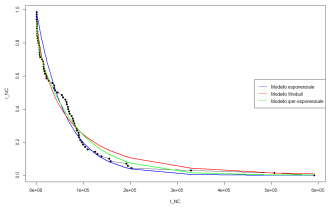
Figura 14: Modelli di reliability per R62-M0



((a)) R63-M1-N0



((b)) R63-M1-N8



((c)) R63-M1-NC

Figura 15: Modelli di reliability per R63-M1

Dall'analisi delle figure 14 e 15 si evince che tutti i modelli approssimano abbastanza bene la reliability. Infatti, i test di Kolmogorov-Smirnov hanno restituito valori di p-value maggiori di 0.05. In particolare, per i nodi R62-M0-N0, R62-M0-N4 ed R62-M0-NC il modello che best-fit è il modello Weibull; invece, per i nodi R63-M1-N0, R63-M1-N8 e R63-M1-NC il modello best-fit è il modello iper-esponenziale.

5.2 Risultati studio reliability a livello di nodo

Node	R62-M0-N0	R62-M0-N4	R62-M0-NC
Number of lines	368	473	370
Coalescence window	2700	2700	2700
Tuple count	53	54	54
MTTF	Mean: 97968.58 Sd: 103508.5 Median: 73476.5 SIQR: 40724.25	Mean: 96119.57 Sd: 103412.3 Median: 71512 SIQR: 47005.5	Mean: 96120.08 Sd: 103287.4 Median: 71512 SIQR: 41054
MTTF 90% C.I.	[73921.46, 122015.7]	[72330.98, 119908.2]	[72360.22, 119879.9]
MTTF 95% C.I.	[69151.63, 126785.5]	[67615.6, 124623.5]	[67650.53, 124589.6]
Exponential model parameters	λ_1: 1.020735e-05 p-value: 0.9723 reject: no	λ_1: 1.040371e-05 p-value: 0.9985 reject: no	λ_1: 1.040365e-05 p-value: 0.9747 reject: no
Weibull model parmeters	λ_1: 1.020735e-05 α: 0.95 p-value: 0.9983 reject: no BEST FIT	λ_1: 1.040371e-05 α: 0.95 p-value: 0.9985 reject: no BEST FIT	λ_1: 1.040365e-05 α: 0.95 p-value: 0.9747 reject: no BEST FIT
Hyperexponential model parameters	λ_1: 1.020735e-05 λ_2: 1.020735e-06 α_1: 0.9 α_2: 0.95 p-value: 0.9723 reject: no	λ_1: 1.040371e-05 λ_2: 1.040371e-06 α_1: 0.9 α_2: 0.95 p-value: 0.9985 reject: no	λ_1: 1.040365e-05 λ_2: 1.040365e-06 α_1: 0.9 α_2: 0.95 p-value: 0.9747 reject: no

Tabella 4: Failures R62-M0

Node	R63-M1-N0	R63-M1-N8	R63-M1-NC
Number of lines	434	422	497
Coalescence window	1200	1200	1200
Tuple count	70	68	71
MTTF	Mean: 74052.1 Sd: 96062.64 Median: 54585 SIQR: 43060.5	Mean: 76261.36 Sd: 98454.73 Median: 54583 SIQR: 42489.5	Mean: 73003.23 Sd: 102956.5 Median: 50271 SIQR: 41404.25
MTTF 90% C.I.	[54767.3; 93336.9]	[56195.1; 96327.6]	[52486.8; 93519.7]
MTTF 95% C.I.	[50975.3; 97128.9]	[52246.4; 100276.4]	[48454.1; 97552.3]
Exponential model parameters	λ_1: 1.350401e-05 p-value: 0.3465 reject: no	λ_1: 1.31128e-05 p-value: 0.5848 reject: no	λ_1: 1.369802e-05 p-value:0.3551 reject: no
Weibull model parmeters	λ_1: 1.350401e-05 α: 0.95 p-value: 0.9589 reject: no	λ_1: 1.31128e-05 α: 0.95 p-value: 0.9535 reject: no	λ_1: 1.369802e-05 α: 0.95 p-value: 0.8787 reject: no
Hyperexponential model parameters	λ_1: 1.350401e-05 λ_2: 1.350401e-06 α_1: 0.2 α_2: 0.8 p-value: 0.9942 reject: no BEST FIT	λ_1: 1.31128e-05 λ_2: 1.31128e-06 α_1: 0.2 α_2: 0.8 p-value: 0.9998 reject: no BEST FIT	λ_1: 1.369802e-05 λ_2: 1.369802e-06 α_1: 0.2 α_2: 0.8 p-value: 0.9614 reject: no BEST FIT

Tabella 5: Failures R63-M1

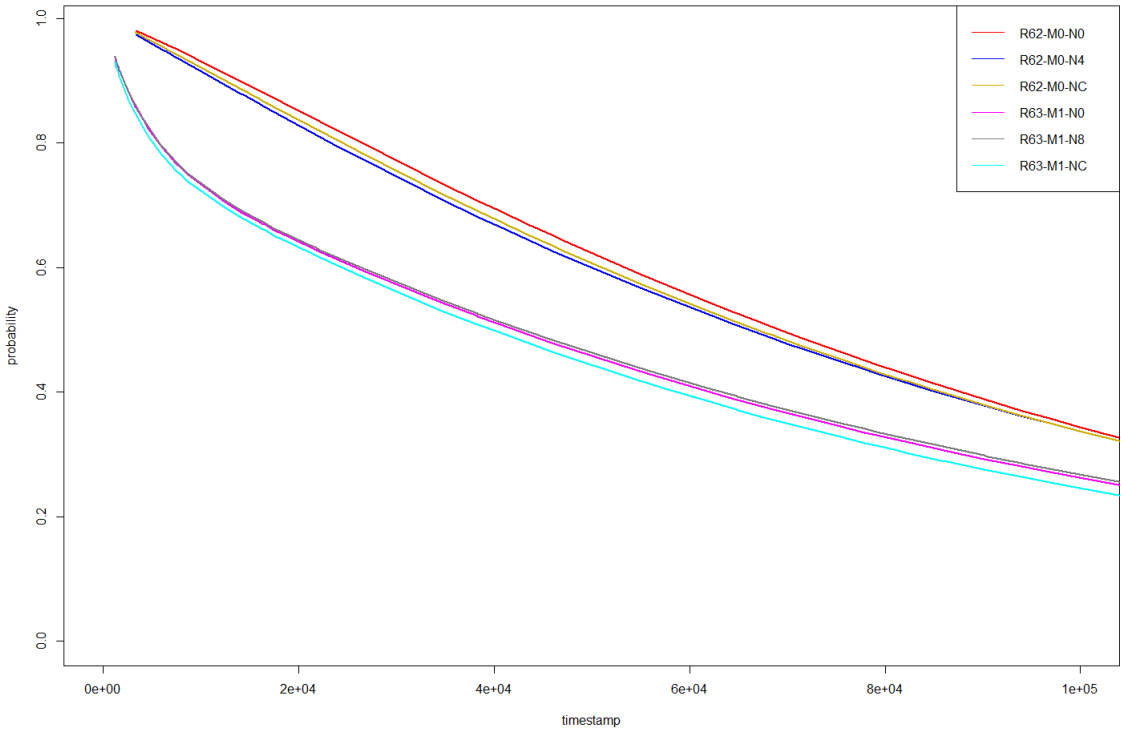


Figura 16: Confronto tra le reliability di tutti i nodi analizzati

Dalla figura 16 è possibile notare come il nodo con MTTF più elevato, e quindi più soggetto a fallimenti, sia: R63-M1-NC; mentre il nodo con MTTF più basso, e quindi meno soggetto a fallimenti, sia: R62-M0-N0. I dati vengono interpretati in questo modo perchè la figura mostra il plot di modelli di reliability, quindi la probabilità che **non** si verifichi un failure.

Appendice A Scelta della finestra

A.1 Settembre

```
# Lettura del dataset tcount-bglsep_1.txt
tcount.bglsep <- read.delim("Homework 1/ffdatools/counts/tcount-bglsep_1.txt")
CWIN<-tcount.bglsep$CWIN
COUNT<-tcount.bglsep$COUNT
p<-plot(CWIN, COUNT, type='o')

# Analisi ravvicinata per scegliere i parametri per trovare la retta tangente
p<-plot(CWIN, COUNT, type='o', xlim =c(0,500), ylim=c(0,500))

# Plot della retta tangente che passa per il punto di ginocchio
x1=c(0,295)
y1=c(295,0)
line<-lines(x1,y1,type='l', col='red', lwd=3)

# lm per ottenere i coefficiente angolare e intercetta della retta tangente
lm(x1 ~ y1)

# Selezione del valore di CWIN nellintorno destro del punto di tangenza
#abline(v=100, lwd=2, col="blue")
point <- points(CWIN[which.max(CWIN > 100)], COUNT[which.max(CWIN>100)], pch=20, col="green", cex=2)
CWIN[which.max(CWIN > 100)] #restituisce il valore di CWIN selezionato
```

A.2 Ottobre

```
# Lettura del dataset tcount-bgloct_1.txt
tcount.bgloct <- read.delim("Homework 1/ffdatools/counts/tcount-bgloct_1.txt")
CWIN<-tcount.bgloct$CWIN
COUNT<-tcount.bgloct$COUNT
p<-plot(CWIN, COUNT, type='o')

# Analisi ravvicinata per scegliere i parametri per trovare la retta tangente
p<-plot(CWIN, COUNT, type='o', xlim =c(0,400), ylim=c(0,400))

# Plot della retta tangente che passa per il punto di ginocchio
x1=c(0,295)
y1=c(195,0)
line<-lines(x1,y1,type='l', col='red', lwd=3)

# lm per ottenere i coefficiente angolare e intercetta della retta tangente
lm(x1 ~ y1)

# Selezione del valore di CWIN nellintorno destro del punto di tangenza
#abline(v=100, lwd=2, col="blue")
point <- points(CWIN[which.max(CWIN > 100)], COUNT[which.max(CWIN>100)], pch=20, col="green", cex=2)
CWIN[which.max(CWIN > 100)] #restituisce il valore di CWIN selezionato
```

Appendice B Analisi degli interarrivi

Il seguente script R è stato utilizzato per svolgere le prime analisi sugli interarrivi. Per semplicità si riporta lo script relativo agli interarrivi del mese di settembre con una finestra pari a 120 secondi. Per analizzare quelli relativi al mese di ottobre bisogna sostituire il path del dataset da leggere.

```
# Lettura del dataset interarrivals di tuples-bglsep_1-120
interarrivals <- read.table("C:/Data Science/Homework
    1/ffdatools/tuples-bglsep_1-120/interarrivals.txt")

# Calcolo della media
media <- mean(interarrivals$V1)
# Calcolo numero degli interarrivi
n <- length(interarrivals$V1)

# Calcolo deviazione standard
deviazione_standard <- sd(interarrivals$V1)
deviazione_standard
# Calcolo mediana
m<-median(interarrivals$V1)

# Calcolo range semi-inter-quartile
quartiles<-quantile(interarrivals$V1, probs=c(0,0.25,0.5,0.75,1))
SIQR<-(quartiles[4] - quartiles[2])/2

# Calcolo errore standard
errore_standard <- deviazione_standard/ sqrt(n)

# Calcolo intervallo di confidenza
#alpha = 0.10 # 90%
alpha = 0.05 # 95%
gradi_di_liberta = n - 1
t_score = qt(p=alpha/2, df=gradi_di_liberta,lower.tail=F)
errore_margine <- t_score * errore_standard

limite_inferiore <- media - errore_margine
limite_superiore <- media + errore_margine
```

Appendice C Modellazione Reliability

C.1 Settembre

```
# Lettura del dataset interarrivals di tuples-bglsep_1-120
interarrivals <- read.table("Homework 1/ffdatools/tuples-bglsep_1-120/interarrivals.txt")

# Plot della ecdf
plot (ecdf(interarrivals$V1), col="blue", main=NULL, xlab="time (s)", ylab="p")

# Calcolo della reliability come 1 - ttf
ttf<-ecdf(interarrivals$V1)
t<-knots(ttf) # knots restituisce i punti su cui la ecdf  stata invocata
r <- 1-ttf(t)

lines(t, r, , type="o", pch=16)
legend( x="right", legend=c("CDF empirica", "Reliability"), col=c("blue","black"), lwd=1)

plot(t, r,type="o", pch=16 )

# Stima delle regressione: modello esponenziale
exp_mod <- nls (r ~ exp(-(l*t)), start=list(l=(1/mean(interarrivals$V1))))
lines(t, predict(exp_mod), col="blue", lwd=2)
ks.test(r, predict(exp_mod))

# Stima delle regressione: modello weibull
wei_mod<-nls (r ~ exp(-(l*t)^a), start=list(l=(1/mean(interarrivals$V1)), a=0.95))
lines(t, predict(wei_mod), col="red", lwd=2)
ks.test(r, predict(wei_mod))

# Stima delle regressione: modello iperesponenziale (1)
hex_mod<-nls (r ~ 0.5*exp(-(l1*t))+0.5*exp(-(l2*t)),
     start=list(l1=(1/mean(interarrivals$V1)),l2=7.282211e-06 ))
lines(t, predict(hex_mod), col="magenta", lwd=2)
ks.test(r, predict(hex_mod))

# Stima delle regressione: modello iperesponenzionale (2)
hex2_mod<-nls (r ~ 0.4*exp(-(l1*t))+0.6*exp(-(l2*t)),
     start=list(l1=(1/mean(interarrivals$V1)),l2=7.282211e-06 ))
lines(t, predict(hex2_mod), col="green", lwd=2)
ks.test(r, predict(hex2_mod)) #restituisce il p-value pi alto.
legend( x="right",
       legend=c("Modello esponenziale", "Modello Weibull", "Modello iper-esponenziale"),
       col=c("blue","red", "green"), lwd=1)
```

C.2 Ottobre

```
# Lettura del dataset interarrivals di tuples-bgloct_1-120
interarrivals <- read.table("Homework1/ffdatools/tuples-bgloct_1-120/interarrivals.txt")

# Plot della ecdf
plot (ecdf(interarrivals$V1), col="blue", xlim=c(0,130000), main=NULL, xlab="time (s)", ylab="p")

# Calcolo della reliability come 1 - ttf
ttf<-ecdf(interarrivals$V1)
t<-knots(ttf) # knots restituisce i punti su cui la ecdf  stata invocata
r <- 1-ttf(t)

lines(t, r, , type="o", pch=16)
legend( x="right", legend=c("CDF empirica", "Reliability"), col=c("blue","black"), lwd=1)

plot(t, r,type="o", pch=16 )

# Stima delle regressione: modello esponenziale
exp_mod <- nls (r ~ exp(-(l*t)), start=list(l=(1/mean(interarrivals$V1))))
lines(t, predict(exp_mod), col="blue", lwd=2)
ks.test(r, predict(exp_mod))

# Stima delle regressione: modello weibull
wei_mod<-nls (r ~ exp(-(l*t)^a), start=list(l=(1/mean(interarrivals$V1)), a=0.95))
lines(t, predict(wei_mod), col="red", lwd=2)
ks.test(r, predict(wei_mod))

# Stima delle regressione: modello iperesponenziale (1)
hex_mod<-nls (r ~ 0.5*exp(-(l1*t))+0.5*exp(-(l2*t)),
     start=list(l1=(1/mean(interarrivals$V1)),l2=4.964024e-06 ))
lines(t, predict(hex_mod), col="magenta", lwd=2)
ks.test(r, predict(hex_mod))

# Stima delle regressione: modello iperesponenzionale (2)
hex2_mod<-nls (r ~ 0.4*exp(-(l1*t))+0.6*exp(-(l2*t)),
     start=list(l1=(1/mean(interarrivals$V1)),l2=4.964024e-06 ))
lines(t, predict(hex2_mod), col="green", lwd=2)
ks.test(r, predict(hex2_mod)) #restituisce il p-value pi alto.

legend( x="right",
       legend=c("Modello esponenziale", "Modello Weibull", "Modello iper-esponenziale"),
       col=c("blue","red", "green"), lwd=1)
```

Appendice D Creazione Transazioni

```
import glob
import sys


def readfile(files):
    """
    Funzione che prende in input tutti i file *.txt presenti nella cartella "tuple" e
    restituisce le transazioni escludendo le ripetizioni dei failures per lo stesso nodo
    :param files: Cartella contenente i file da cui estrarre le transazioni
    :returns: File contenente le transazioni
    """
    final = []
    for path in files:
        f=open(path, "r")
        nodes = []
        temp = []
        for line in f:
            temp.append(line.split(" ")[2])
        nodes = list(dict.fromkeys(temp))
        final.append(nodes)
    with open(folder+"/output.csv", mode="w") as file:
        for item in final:
            row = (', '.join(item))
            file.write(row+"\n")
    print("\nOUTPUT: ffdatools/"+folder+"output.csv")


if __name__ == '__main__':
    args = len(sys.argv)
    print(sys.argv[1])
    if len(sys.argv)>2:
        print ("\nERROR: incorrect number of arguments")
    else:
        folder = sys.argv[1]
        files = glob.glob(folder + "/*.txt")
        files.remove(folder+"\\interarrivals.txt")
        readfile(files)
```

Appendice E Itemset Frequenti

E.1 Settembre

```
library('arules')
library('arulesViz')

#lettura delle transazioni del mese di settembre
data_sep <- read.transactions("/Homework1/ffdatools/tuples-bglsep_1-120/output.csv", sep=",")

#calcolo della soglia: almeno due fallimenti in un giorno per il mese di settembre
s_sep<-60/length(data_sep)

#calcolo dei frequent itemset
freq_sep<-apriori(data_sep, parameter =list(support=s_sep, confidence=0.75, minlen=2, maxlen=20,
    target="frequent itemsets"))

#calcolo delle regole associative
rules_sep<-apriori(data_sep, parameter =list(support=s_sep, confidence=0.75, minlen=2, maxlen=20,
    target="rules"))
summary(rules_sep)
summary(freq_sep)
inspect(freq_sep)
inspect(sort(rules_sep, by="confidence"))

plot(rules_sep, method = "grouped", control = list(k = 10), col=rainbow(3))
```

E.2 Ottobre

```
##lettura delle transazioni del mese di ottobre
data_oct <- read.transactions("Homework1/ffdatools/tuples-bgloct_1-120/output.csv", sep=",")

#calcolo dei frequent itemset
freq_oct<-apriori(data_oct, parameter =list(support=0.257, confidence=0.75, minlen=2, maxlen=20,
    target="frequent itemsets"))

#calcolo delle regole associative
rules_oct<-apriori(data_oct, parameter =list(support=0.257, confidence=0.75, minlen=2, maxlen=20,
    target="rules"))

summary(freq_oct)
summary(rules_oct)

nonr_rules <- rules_oct[!is.redundant(rules_oct)] #rimozione regole ridondanti

summary(nonr_rules)
inspect(nonr_rules)

options(digits=3)
plot(nonr_rules, method = "grouped", control = list(k = 50), col=rainbow(3))
```

Appendice F Selezione dei nodi

```
import glob
import sys

def selection(file, node):
    """
    Funzione che seleziona un nodo specifico e ne cerca
    all'interno di un file tutte le occorrenze
    :param file: File unito in cui selezionare i nodi
    :param node: Nome del nodo specifico da selezionare
    :returns: File contente i failure del solo nodo selezionato
    """
    temp = []
    with open(file, mode="r") as file:
        for row in file:
            if node in row:
                temp.append(row)

    with open(node, mode="w") as output:
        for item in temp:
            output.write(item)
    print("\nOUTPUT: ffdatools/"+node)


if __name__ == '__main__':
    args = len(sys.argv)
    if len(sys.argv)>3:
        print ("\nERROR: incorrect number of arguments")
    else:
        file = sys.argv[1]
        node = sys.argv[2]
        selection(file, node)
```