

**UNIVERSITÀ DEGLI STUDI DI
NAPOLI FEDERICO II**

Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica



tesi di laurea specialistica

Caratterizzazione del workload per l'analisi quantitativa del “Software Aging”

Anno Accademico 2010/2011

relatore
Ch.mo prof. Domenico Cotroneo

correlatore
Ing. Antonio Bovenzi

candidato
Elisabetta Traino
matr. 885/417

Caratterizzazione del workload per l'analisi quantitativa del Software Aging

Elisabetta Traino

15 dicembre 2011

*'Ora volerai, Fortunata. Respira. Senti la pioggia. È acqua. Nella tua vita avrai molti motivi per essere felice, uno di questi si chiama acqua, un altro si chiama vento, un altro ancora si chiama sole e arriva sempre come una ricompensa dopo la pioggia.
Senti la pioggia. Apri le ali.'*

'La pioggia. L'acqua. Mi piace!'

'Ti voglio bene. Sei un gatto molto buono.'

'Ora volerai, il cielo sarà tutto tuo.'

'Vola!'

'Volo! Zorba! So volare!'

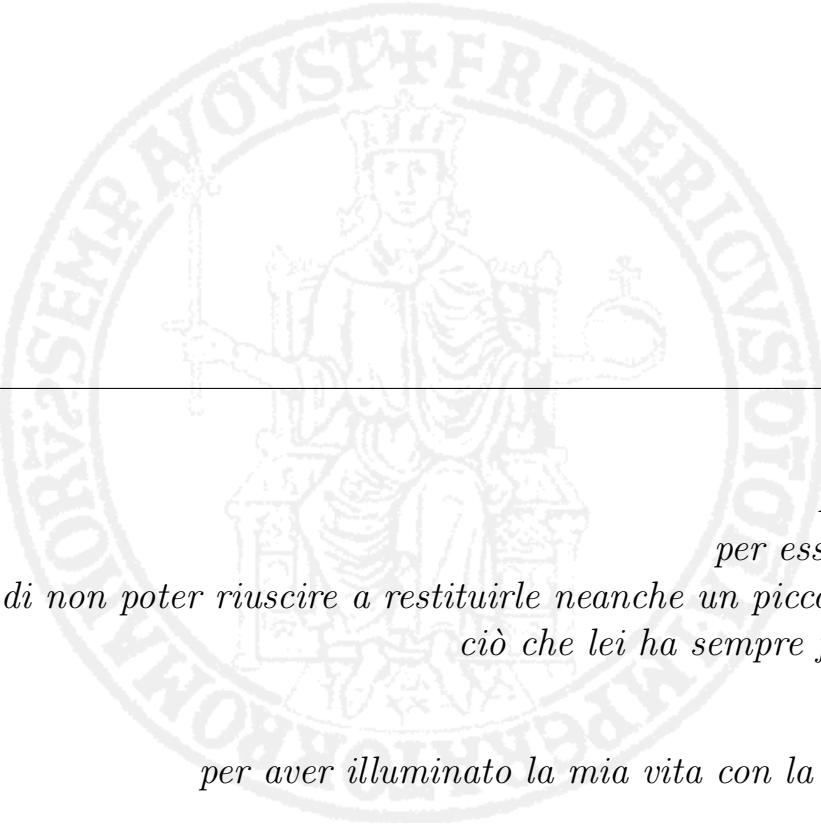
'Bene, gatto. Ci siamo riusciti'

'Si, sull'orlo del baratro ha capito la cosa più importante.'

'Ah si? E cosa ha capito?'

'Che vola solo chi osa farlo.'

(Luis Sepúlveda)



*A mia madre,
per essere ciò che è.*

*Credo di non poter riuscire a restituirlle neanche un piccolo pezzetto di
ciò che lei ha sempre fatto per me.*

*A Lelio,
per aver illuminato la mia vita con la mano di Dio.*

Indice

Ringraziamenti	iv
Introduzione	1
1 Il problema del software aging	5
1.1 Software Dependability	5
1.2 Software Faults	6
1.2.1 Classificazione dei Software Faults	7
Classificazione ODC	8
Classificazione Gray	9
1.3 Analisi della letteratura	11
1.3.1 Software Rejuvination	12
1.3.1.1 Approcci di modellazione analitica	13
1.3.1.2 Approcci basati su misure sperimentali	14
1.4 Il ruolo del workload	14
1.4.1 Benchmarking	15
1.4.2 Predittori online	16
2 Caratterizzazione del workload	17
2.1 Caratterizzazione ad alto livello	18
2.2 Progettazione degli esperimenti	19
2.2.1 Principi base del Design of Experiment	19
2.2.2 La metodologia adottata	20
2.2.3 Tipologie di DOE	22
2.2.4 Classificazione del DOE in base ai fattori	24
2.2.4.1 Disegno completamente casualizzato	24
2.2.4.2 Disegno casualizzato a blocchi	25
2.2.4.3 Disegno multifattoriale	26
2.2.5 Determinazione del piano sperimentale	26
2.3 Caratterizzazione dipendente dall'applicazione	29
2.3.1 Analisi dei cluster	29
2.3.1.1 Preprocessing dei dati	30
Data standardization	31
Data correlation	31
Algoritmo k-means	31
2.3.2 Test preliminari	32
2.3.2.1 Capacity test	33
2.3.2.2 Test zero	33
2.4 Rilevazione e stima dei trend statistici	34

2.4.1	Il test Mann Kendall	35
2.4.2	La stima della pendenza di Sen	35
3	Studio di un caso	37
3.1	MySQL	37
3.1.1	Specificazione del Workload	38
3.1.2	Benchmark di DBMS	39
3.1.2.1	Motivazioni	40
3.1.2.2	Il benchmark TPC-E	41
3.1.2.3	Struttura database del benchmark	42
3.1.2.4	Passi per l'utilizzo del benchmark	43
3.1.2.5	Le Query TPC-E considerate	44
4	Analisi dei risultati sperimentali	45
4.1	Descrizione del Testbed	45
4.2	Raccolta dei dati	46
4.2.1	Monitoring	46
4.2.2	Filtraggio e preprocessing dei risultati	48
4.3	Esecuzione dei test preliminari	48
4.3.1	Capacity test	48
4.3.1.1	Determinazione dei valori dei livelli	49
Livelli intensità	49	
Livelli size	49	
Livelli tipologia di richiesta	50	
Livelli variazione della richiesta	52	
Generazione del traffico per il DBMS Mysql	52	
4.3.2	Test zero	55
4.4	Esecuzione esperimenti	57
4.5	Analisi dei risultati	58
4.5.1	Preprocessing delle variabili monitorate	58
4.5.2	Principal Component Analysis	60
4.5.2.1	Risultati dell' analisi	68
4.5.3	Determinazione dei trend di Aging	71
4.5.4	Valutazione dei trend osservati rispetto alle caratteristiche del workload	75
4.5.5	Stima del Time To Exaustion (TTE)	79
5	Conclusioni e Sviluppi Futuri	
A	Struttura e Script per il Database TPC-E	81
A.1	Schema ER TPC-E	81
A.2	Script per il caricamento del database	83
A.3	Script per la creazione degli indici del database	84
A.4	Script per la creazione delle chiavi	84
Appendici		81
B	Query TPC-E utilizzate	86

B.1	Query per la size LOW	86
B.2	Query per la size High	89
C	Significato variabili monitorate	93
D	MySQL Traffic Generator	95
D.1	Class Worker	95
D.2	Class Thread Generator	99
Bibliografia		105

Elenco delle tabelle

2.1	Esempi di applicazioni industriale del DoE	20
2.2	Piano sperimentale	29
4.1	Cross-correlazione tra le variabili candidate alla clusterizzazione	51
4.2	Clusters delle richieste	52
4.3	Piano sperimentale per MySQL	57
4.4	Piano sperimentale per MySQL - valori specifici utilizzati	57
4.5	Cross-Correlazione Tra le variabili scelte per il monitoraggio con Corr > 0.90	59
4.6	Cross-Correlazione Tra le variabili scelte per il monitoraggio con Corr > 0.70 e < 0.90	59
4.7	Valori della varianza spiegata	65
4.8	Coefficienti analisi PCA delle prime cinque componenti	66
4.9	Risultati Mann-Kendall test - Esperimento 1 - Variabili relative allo stato della memoria	72
4.10	Risultati Mann-Kendall test - Esperimento 2 - Variabili relative allo stato della memoria	73
4.11	Risultati Mann-Kendall test - Esperimento 3 - Variabili relative allo stato della memoria	73
4.12	Risultati Mann-Kendall test - Esperimento 4 - Variabili relative allo stato della memoria	74
4.13	Risultati Mann-Kendall test - Esperimento 1-2-3-4 - Variabili relative alle prestazioni del sistema	74
4.14	TTE (in ore (a), giorni (b)) relativi alle variabili MemFree e Active per tutti gli esperimenti	79
5.1	Variabili da monitorare	
C.1	Tabella Variabili monitorate numero 1	93
C.2	Tabella Variabili monitorate numero2	93
C.3	Tabella Variabili monitorate numero 3	94

Elenco delle figure

1.1	Propagazione guasto - errore - fallimento	7
2.1	Rappresentazione di un sistema sperimentale	21
2.2	Esempi di trend statistici	34
4.1	Test-bed utilizzato per gli esperimenti	47
4.2	MySQL - Capacity test - Size low	54
4.3	MySQL - Capacity test - Size high	55
4.4	Test zero - Memoria libera del sistema	56
4.5	Test zero - Resident Set Size del processo	57
4.6	Esempio di variazione di punti di vista per massimizzare l'informazione visibile su un fenomeno. Immagine tratta da [1]	60
4.7	Esempio di cambio di riferimento mediante PCA	61
4.8	Esempio di dataset	61
4.9	Dispersione delle variabili monitorate	64
4.10	Grafico di Pareto della varianza spiegata	66
4.11	Analisi PCA - coefficienti delle variabili monitorate e osservazioni Z- trasformate dei dati nel nuovo riferimento	67
4.12	Relazione dati nel nuovo riferimento - ore trascorse per la prima componente.	68
4.13	Relazione dati nel nuovo riferimento - ore trascorse per la seconda componente.	69
4.14	Trend rilevati per le variabili monitorate	75
4.15	Trend rilevati per le variabili monitorate	76
4.16	Trend della variabile Buffers	77
4.17	Trend rilevati per le variabili monitorate - indicatori delle prestazioni del sistema	78
A.1	Schema ER delle Tabelle utilizzate in TPC-E	82

Ringraziamenti

Finalmente è arrivata una delle parti che più desideravo scrivere, da tempo.

Chi mi conosce bene sa quanto sia importante per me ringraziare e dimostrare l'affetto alle persone che amo.

Il primo ringraziamento, oltre che la dedica, non soltanto di questo lavoro ma di tutto il mio percorso di studi, dal primo giorno all'ultimo, va a MIA MADRE. Una donna che nonostante tutti i brutti i momenti, riesce sempre a guardare la vita con un sorriso, trasmettendo questo a tutte le persone che le sono accanto. A lei devo i numerosi sacrifici che ha fatto per me, che già da tempo mi pesano nel cuore. Rappresenta per me un grande esempio, come donna e come mamma. Spero con tutto il cuore un domani di poterle donare ciò che merita, farò di tutto per riuscirci.

Il secondo particolare ringraziamento è a Lelio, la mia luce nell'oscurità. Mi ha cambiato la vita e mi ha fatto capire quanto una persona può essere davvero speciale. Fin dal primo momento ha sempre cercato di migliorarmi, come ingegnere e come donna. In ogni momento ha sempre capito le mie sofferenze e ha cercato di accontentarmi in ogni modo. Oggi se sono una persona e un ingegnere migliore lo devo a lui. Ha cambiato il mio modo di apprendere, il mio modo di approcciarmi alla vita. Senza di lui non sarei la persona che sono ora e non avrei accanto a me un animo così fuori dal comune.

Un altro ringraziamento va a mia sorella con cui, dopo qualche anno di piccole incomprensioni mi sono legata più di prima. Anche se ha un modo tutto suo, sa starmi accanto in ogni momento con il cuore e crede in me, da sempre.

A mio padre devo la realizzazione di questo sogno e l'avermi sempre supportata.

Ai miei zietti Rosaria e Carlo va un ringraziamento enorme, con il cuore. Hanno sempre creduto in me e sono i miei secondi genitori. Vi adoro! Grazie per esserci SEMPRE!

Un ringraziamento particolare va alla famiglia Di Martino, per avermi accolto come una figlia e per essermi sempre stata accanto e incoraggiata, in particolare ad Anna per l'enorme disponibilità, complicità e la dolcezza mostratami. A Daniela, semplicemente....perché è una cognata adorabile ed una seconda sorella per me :)

Ai nonni Marchesano perché sono troppo dolci con me e perché mi hanno aperto il loro cuore senza riserve. Per me è una grande felicità. Ringrazio zia Rossella, zio Pio, Nancy e soprattutto Ida perché mi hanno sempre fatto sentire a casa e voluta tanto bene.

Alle mie migliori amiche, Tiziana e Paola, un enorme GRAZIE per aver vissuto momenti divertentissimi e spensierati insieme all'intensa complicità femminile, tra una 'festa' e l'altra :) 'sognando' Sex and City :) ..siete e sarete per me come delle sorelle; siete state il MIO bellissimo momento di svago in molte situazioni che porterò sempre nel cuore.

Un particolare ringraziamento va ad Emilio, che è sempre stato più che disponibile con la mia famiglia senza mai tirarsi indietro.

Questi mesi di lavoro sono stati splendidi, soprattutto grazie alle persone che mi hanno seguito in maniera molto attenta. Ringrazio Antonio, per aver tirato la parte migliore di me e per avermi insegnato un sacco di trucchetti :) Ringrazio il prof. Cotroneo perché la sua enorme esperienza ha segnato la mia preparazione e per il suo modo di insegnare che è eccezionale. La sua sensibilità nel capire le persone è fuori dal comune. Nonostante il mio caratteraccio :) è riuscito a farmi comprendere, nella maniera più serena possibile, alcuni aspetti importanti della vita. A lui un ringraziamento molto molto sentito, con il cuore.

Un ringraziamento particolare va anche al mitico MOBILAB per l'enorme disponibilità e gentilezza. Siete un gruppo di persone davvero speciali!

In questa fase conclusiva ci tengo molto a ricordare i professori che in tutto il mio percorso di studi, dalla triennale alla specialistica, hanno lasciato un segno.

Mi ricorderò dei proff. De Bernardo e De Vito di elementi di automazione e geometria ed algebra lineare per la serenità e l'enorme chiarezza delle lezioni e spiegazioni.

Il prof. Stefano Russo per i fantastici ragionamenti astratti sui sistemi distribuiti e per avermi insegnato l'importanza del metodo scientifico. A volte anche solo una correzione di poche parole ti rimane impressa per tutta la vita, facendoti cambiare il modo di affrontare i problemi.

Il prof. Esposito per avermi insegnato la risoluzione pratica di complessi problemi di ingegneria.

Le lezioni tenute dal prof. Mazzeo che erano per me di enorme stimolo intellettuale, spaziando tra diversi argomenti scientifici oltre che sull'importanza dell'"imparare a ragionare come degli ingegneri".

Mi ricorderò del prof. Mazzocca per il suo essere così gentile con tutti e per avermi fatto appassionare a tutti i ragionamenti alla base delle architetture dei calcolatori.

Un ringraziamento a questa mia splendida Università e a tutti i professori che ogni giorno lavorano motivati da un'enorme passione. Devo ammettere che, potrà sembrare strano per qualcuno, ma tutto questo mi mancherà!

Ringrazio i miei carissimi amici, Manuele, Carmen, Fabrizio e il mio cognatino Luigi per avermi regalato splendidi momenti di svago e relax.

Anche se in maniera MOLTO particolare ringrazio la mia Chérie, per avermi fatto compagnia tutte le volte che studiavo, dietro la mia sedia o anche sulle mie gambe:) Ti amo cucciolina mia.

In ultimo, ma non come importanza, ringrazio me stessa. Perché nei momenti bui, quando ero completamente sola e fortemente scoraggiata, dopo alcuni momenti di crisi e smarrimento, sono riuscita sempre a ritrovare la mia strada e a credere in me stessa. Se non fosse stato così, i numerosi ostacoli che ho incontrato, insieme alla mia fragilità, non avrebbero reso possibile tutto questo. Per cui....si! UN 'GRAZIE' ED UNA DEDICA SOPRATTUTTO A ME! :) ..che, anche se a volte ho avuto qualche dubbio, facendomi prendere dal grigiore, mi sono rialzata con tutta la mia forza e ho raggiunto serenamente e splendidamente il mio obiettivo.

Napoli, Italy

Elisabetta

Data

Introduzione

E' ben noto che i guasti software rappresentano la principale causa di fallimento nei moderni sistemi informatici [2, 3, 4]. Per guasto software si intende uno stato improprio del software causato da errori di progettazione.

Una grossa percentuale di fallimenti software è dovuta al Software Aging [5]. Si tratta di un fenomeno, osservato empiricamente in numerosi sistemi software, per il quale il sistema degrada con il tempo [5]. In particolare, un fallimento può assumere la forma di un servizio totalmente o parzialmente incorretto (es. risultati errati o graduale aumento dei tempi di risposta), hang e/o crash del sistema. Cause tipiche di questo fenomeno sono dovute all'accumulazione di errori numerici, per esempio errori di *round-off* (come nel famoso caso del fallimento del sistema missilistico Patriot nel 1991 [5]), corruzione dei dati, porzioni di memoria non correttamente deallocate, esaurimento delle risorse del sistema operativo [6].

Per contrastare il fenomeno del Software Aging Huang et.al [7] hanno proposto una tecnica chiamata 'Software Rejuvination'. Essa consiste nel i) terminare saltuariamente il sistema software; ii) eliminare le condizioni di errore accumulate; iii) riportare il sistema in uno stato interno corretto. Esempi di ripristino dello stato corretto sono la garbage collection, il flushing delle tabelle del kernel del sistema operativo e la reinizializzazione delle strutture dati [7, 8].

In letteratura si utilizzano due differenti approcci nello studio del Software Aging, l'*analytical modeling* e il *measurement based*. I primi si basano sull'utilizzo di modelli analitici, tipicamente reti di Petri stocastiche o catene di Markov [9], per rappresentare

il degrado del sistema. La risoluzione di questi modelli ha l'obiettivo di ottenere la schedulazione ottima per l'attività di *Software Rejuvination*. L'approccio basato sulle misure utilizza tecniche statistiche ben note per la rilevazione e validazione dell'esistenza di un particolare fenomeno. Nel caso del Software Aging non possono essere utilizzate le tradizionali tecniche di misure per la valutazione dell'affidabilità, a causa della modalità con cui tale fenomeno si presenta. L'idea principale è monitorare periodicamente il sistema, collezionare i dati sulle variabili indicative della salute del software in esecuzione e applicare tecniche di rilevazione dei trend statistici per individuare e validare l'esistenza di aging utilizzando la metrica del TTE (Time To Exhaustion) [6]. Siccome l'approccio analitico richiede un'approfondita conoscenza del sistema mentre quello basato sulle misure fornisce una stima piuttosto affidabile del tempo di esaurimento delle risorse, ma talvolta non rende possibile la determinazione di una schedulazione ottima per la *Software Rejuvenation*. Per tali motivi risulta sempre più frequente l'utilizzo di approcci 'ibridi' che combinano le suddette tecniche [10] [11]. In questi ultimi studi, la stima del TTE può essere valutata in maniera dipendente [10] o indipendente dal workload [6], cioè il carico sottoposto al sistema. Intuitivamente il TTE è influenzato dal workload applicato e dunque differenti workload potrebbero mostrare differenti dinamiche di aging. L'approccio indipendente dal workload non considera questo aspetto e può dare una stima affidabile del TTE solo sotto l'assunzione di workload. Lo scopo dell'analisi di aging basata sul workload è stimare la correlazione tra quest'ultimo, lo stato di esecuzione e la disponibilità del sistema.

Diversi lavori hanno indagato sulla dipendenza del Software Aging dal workload [12, 11, 10, 13, 14] confermando che le dinamiche di aging sono relazionate agli stati di workload e che *la metodologia workload-driven risulta essere molto più precisa per modellare il fenomeno del Software Aging* [10]. Tuttavia, gli indicatori di workload utilizzati in suddetti studi rappresentano parametri interni (cioè a livello di sistema) e per questo sono fattori difficilmente controllabili [13].

Un lavoro indirizzato verso l'impatto di workload a livello applicativo sulle dinamiche di aging è presentato in [15]. Questo lavoro si focalizza su un'applicazione spe-

cifica, Apache Web Server, quindi considerando dei parametri di workload specifici dell'applicazione.

In tutti i casi citati vengono presi in considerazione parametri di workload specifici del sistema o della particolare applicazione in analisi, non rendendo possibile un confronto tra risultati ottenuti per sistemi diversi [13].

Nel presente lavoro di tesi viene analizzato il legame tra il Software Aging e il workload. Lo scopo è quello di analizzare quali fattori del Workload, indipendenti dall'applicazione, influenzino statisticamente il fenomeno del Software Aging. Il metodo è basato su un processo di caratterizzazione del workload che, partendo da una descrizione ad alto livello, conduce alla progettazione di un insieme di esperimenti da eseguire su una o più applicazioni da esaminare. Tale caratterizzazione consiste nel determinare un insieme di parametri indicativi del workload, indipendenti dall'applicazione, identificati in *intensità*, *dimensione* del workload, *tipo* e *variazione* del tipo di richiesta.

Il metodo proposto viene applicato nel presente lavoro di tesi al Database Management System (DBMS) MySQL. A tal fine, il sistema considerato viene inizialmente analizzato mediante test di capacità (*capacity test*) per determinare il limite dell'applicazione testata e per scegliere valori significativi per i parametri del workload dipendenti dall'applicazione. Tali esperimenti consistono nel determinare una metrica per misurare il throughput del sistema e quindi nel sollecitare il sistema con un carico di intensità crescente fino al raggiungimento del limite di capacità, al di là del quale il sistema non è più in grado di servire alcuna richiesta. Tale limite viene considerato come rappresentativo del massimo livello di carico gestibile dal sistema. Utilizzando livelli di carico inferiori a quello limite si è certi che il sistema non presenti fallimenti per sovraccarico ma eventualmente fallimenti causati dal Software Aging. I test di capacità sono stati condotti su un test-bed ad-hoc attraverso sia il tool open-source *JMeter*, sia attraverso un generatore di carico appositamente realizzato, basato sullo studio di alcuni benchmark disponibili in letteratura.

Una volta scelti i parametri dipendenti dall'applicazione e identificati i limiti superiori e inferiori per il workload, la durata degli esperimenti viene scelta utilizzando l'appro-

cio proposto in [13], chiamato *test-zero*. In particolare, il *test zero* mira ad identificare la presenza di un trend di aging quando al sistema viene sottoposto il workload meno stressante, analizzando una opportuna variabile osservata durante l'esecuzione del sistema, identificativa di uno stato degradato del sistema, chiamata indicatore di aging [5]. In questa tesi, vengono scelti diversi indicatore di aging tra cui l'occupazione relativa di memoria (memory depletion) [8, 12] e il throughput loss.

La campagna sperimentale condotta dimostra che la metodologia proposta si presta ad essere utilizzata su diversi sistemi e in tal modo consente di comparare i sistemi dal punto di vista dell'aging. La metodologia esposta in questo lavoro può essere usata per scopi di predizione, per esempio per migliorare soluzioni basate sui modelli o sviluppare tool per la stima di aging. In particolare un tool per la predizione al runtime del TTE che, sulla base dei parametri di workload definiti, monitora il workload a tempo di esecuzione e valuta i trend di aging utilizzando modelli dipendenti dal workload.

Il lavoro si articola in quattro capitoli di cui nel seguito si fornisce una breve descrizione.

Il *Capitolo 1* introduce brevemente i concetti di *Software Dependability*, *Software Faults* con particolare approfondimento al fenomeno del *Software Aging*. Segue l'analisi della letteratura in tale settore di ricerca, gli approcci attualmente adottati e la tecnica della *Software Rejuvination*. Il capitolo si conclude con la descrizione del problema del *Workload* e le nuove sfide in questa direzione.

Il *Capitolo 2* descrive l'approccio adottato nella tesi ed in particolare si sofferma sulla caratterizzazione del *Workload* ad alto livello, sulla progettazione degli esperimenti e sulla descrizione del *Workload System-dependent*.

Il *Capitolo 3* presenta il nuovo caso di studio: il DBMS Mysql, motivandone la scelta e descrivendo accuratamente le modalità di progettazione, implementazione del caso di studio e la tecnica utilizzata per la caratterizzazione del *Workload* per tale sistema.

Il *Capitolo 4* presenta e analizza i risultati sperimentali e conclude la tesi con una descrizione di possibili sviluppi futuri.

Senza un avversario la virtù
marcisce.

Seneca

Capitolo 1.

Il problema del software aging

Questa sezione presenta un insieme di definizioni formali riguardanti la Software Dependability, il contesto in cui nasce questo lavoro di tesi e quali sono le problematiche principali secondo cui viene studiata la Software Reliability. Viene poi introdotta la classificazione dei faults e il fenomeno del Software Aging. In seguito all'analisi della letteratura vengono approfonditi gli approcci adottati per tale fenomeno ed in particolare quello maggiormente utilizzato ad oggi, la software rejuvenation. Segue una descrizione delle nuove sfide adottate con particolare approfondimento all'approccio basato sul workload da sottoporre al sistema. Il capitolo si conclude con i possibili scenari che possono essere esplorati con questo tipo di approccio.

1.1 Software Dependability

Per *sistema* si intende una qualsiasi entità in grado di interagire con altre entità, siano esse altri sistemi o utenti umani.

L'insieme di entità con cui un sistema può interagire è rappresentato dal *ambiente*.

Il *comportamento* di un sistema è l'insieme delle azioni che esso compie per implementare le funzioni richieste ed è costituito da una successione di stati.

Il *servizio* fornito da un sistema è il suo comportamento così come viene percepito da un utente, cioè da una qualsiasi altra entità che usufruisce del servizio stesso.

Un servizio si dice corretto se è **conforme** alle specifiche del sistema stesso.

L'*interfaccia* di un sistema è il confine dello stesso percettibile da un utente.

In [16] Laprie definisce **Software Dependability** come '*...the ability to avoid service failures that are more frequent and more severe than acceptable*'. Il concetto di dependability ingloba un insieme di attributi di qualità che assumono singolarmente

un'enfasi riguardante la natura del sistema cui si riferiscono e al particolare contesto applicativo:

- Availability: probabilità che un sistema sia pronto in un certo istante t . In formule: $A = P(\text{!Failure at } t)$

- Reliability: è la misura della continuità di un servizio ovvero la misura dell'intervallo di tempo in cui viene fornito un servizio corretto. In formule:

$$R(t) = P(\text{!Failure in}(0, t))$$

- Safety: l'assenza di condizioni di funzionamento che possano portare il sistema a danneggiare gli utenti e/o l'ambiente in cui opera;

- Perfomability: è metrica introdotta per valutare le performance del sistema anche a valle dell'occorrenza di un fallimento, è utile a sottolineare il suo legame con aspetti sia di PERFORMANCE evaluation sia di dependABILITY.

- Maintenability: la capacità di un sistema di poter essere sottoposto a modifiche e/o riparazioni

- Security: assenza di manipolazioni improprie e accessi non autorizzati al sistema.

Più in dettaglio un sistema sicuro è un sistema in cui coesistano più attributi:

- availability
- confidentiality
- integrity

1.2 Software Faults

Quando un sistema non è in grado di fornire un servizio corretto si parla di fallimento del servizio o semplicemente di fallimento. Siccome un servizio è rappresentato da una sequenza di stati esterni del sistema (percettibili dall'utente), quando si verifica un fallimento significa che almeno uno di tali stati *devia da uno stato corretto*. Tale

deviazione è chiamata *errore*. La causa di tale errore è il *guasto*. Osservando l'immagine 1.1, è possibile fornire importanti definizioni su guasto, errore, fallimento di un sistema.

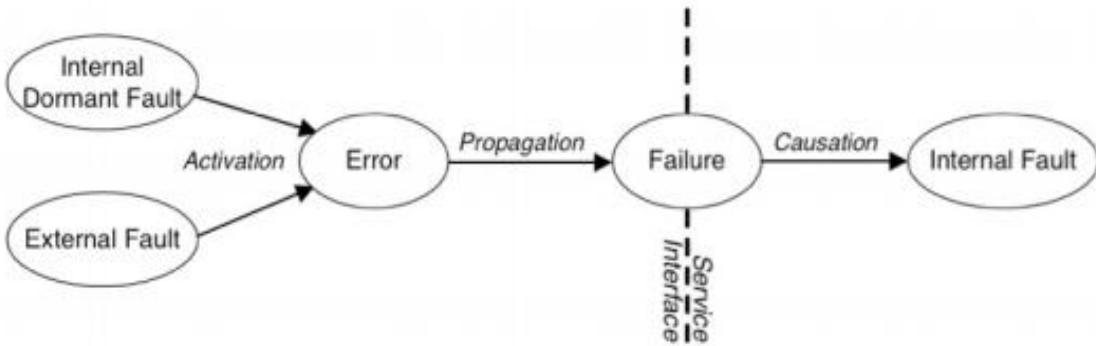


Figura 1.1: Propagazione guasto - errore - fallimento

Si dice che un guasto è attivo quando esso produce un errore, in caso contrario si dice dormiente. Quando un errore si propaga all'interfaccia dell'utente si verifica un fallimento. Nel caso in cui tale errore non porti ad un fallimento si parla di errore latente.

Nonostante esistano diversi tipi di guasti, quelli software rappresentano la causa principale di fallimenti nei moderni sistemi informatici [3, 4]. Infatti le tecniche di fault tolerance, fault avoidance e la modellazione dell'availability e reliability nel caso di guasti hardware risultano molto ben sviluppati rispetto al caso dei guasti software.

1.2.1 Classificazione dei Software Faults

Laprie et al. hanno classificato i Software Faults sulla base di diverse dimensioni, come la fase di creazione o occorrenza (fase di sviluppo o operazionale), gli obiettivi (maliziosi o non maliziosi), l'intento (intenzionale, non intenzionale), capacità (accidentale o incompetenza), persistenza (permanente o transiente). I guasti possono verificarsi in fase operazionale, e in tal caso possono essere permanenti o transienti, o in fase di sviluppo, in questo caso si tratta spesso di guasti permanenti o comunemente chiamati *Software*

Defects. Le tipologie di guasti software possono essere classificati in diversi modi a seconda delle caratteristiche che si desidera enfatizzare.

Classificazione ODC Dal punto di vista della loro semantica, i guasti posso essere classificati secondo lo schema Orthogonal Defect Classification (ODC). Si tratta di un approccio scientifico, utilizzabile per un gran numero di sistemi che fornisce misurazioni molto utili per il recupero di interessanti informazioni in un processo di sviluppo e cioè:

- Tecnica di prevenzione dai difetti o anche detta *defect prevention*. Si tratta di un processo utilizzato per migliorare la qualità del software.
- Metodologia per caratterizzare i difetti software e relazionarli a difetti dei processi.

I concetti base della ODC sono quindi i difetti software e i triggers. I difetti sono appunto imperfezioni presenti nel codice sorgente dei programmi e nella classificazione ODC vengono raggruppati in tipologie ortogonali e non sovrapposte. Con trigger s'intende qualsiasi evento che permette ad un guasto di propagarsi e diventare fallimento. Numerosi triggers possono riguardare lo stesso guasto. In [17] sono definite le seguenti tipologie di difetti:

- Function
- Interface
- Assignment
- Checking
- Timing/Serialization
- Algorithm
- Build/package/merge
- Documentation

Le categorie di trigger più utilizzate sono:

- Boundary Conditions
- Bug Fix
- Recovery
- Exception Handling
- Timing
- Workload

Classificazione Gray La precedente classificazione risulta molto utile dal punto di vista del miglioramento dei processi, provvedendo a fornire informazioni circa le imperfezioni dei sistemi software presenti in fase di sviluppo. Poiché risulta molto costoso rilevare guasti nella fase operazionale rispetto a quella di sviluppo e testing, negli ultimi anni gli ingegneri hanno rivolto i loro sforzi verso la rilevazione e rimozione dei difetti software durante la *fase di sviluppo* attraverso sempre più sofisticate strategie di testing. Azioni condotte sistematicamente e operazioni di testing sul sistema giocano un ruolo importante nel rivelare guasti che eventualmente possono provocare fallimenti durante la fase di esecuzione del software. Ad ogni modo, isolare la causa responsabile dei fallimenti osservati diventa molto complesso se i fallimenti non sono riproducibili. Infatti, l'attivazione dei guasti attraverso l'esecuzione di parte specifiche del software, in cui esso è presente, non è detto che provochi sicuramente un fallimento. Ad esempio, per la propagazione guasto-errore-fallimento prima descritta 1.1, un guasto presente in un algoritmo può portare ad elaborazioni errate per i valori di una particolare variabile utilizzata. Il software può utilizzare tali risultati incorretti per effettuare altri calcoli, con un conseguente accumulo di errori. Il fallimento si manifesta solo quando il sistema usa *almeno uno di questi calcoli incorretti* in modo da influenzare un comportamento del sistema *percettibile* dall'utente o quando la propagazione degli errori provoca *l'occorrenza di un fallimento*.

Sulla base delle relazioni tra guasti, errori e fallimenti è possibile fornire due spiegazioni sul perché un software possa esibire diversi comportamenti a partire da identiche condizioni. Se il ritardo tra l'attivazione del guasto e l'istante di occorrenza del fallimento è troppo grande, risulta molto complesso identificare le azioni da eseguire per riattivare il guasto e causare il fallimento. Infatti, ripetere i passi eseguiti un attimo prima dell'occorrenza del guasto potrebbe non essere sufficiente alla sua riproduzione, dato che potrebbero essere attraversati diversi *stati di errore* nel processo di propagazione degli stessi. Inoltre, altri componenti del sistema indicati complessivamente con *ambiente interno del sistema*, come sistema operativo, altre applicazioni presenti o i componenti hardware, possono influenzare il comportamento del guasto nella specifica applicazione. In sintesi un guasto può portare il software a esibire un comportamento caotico e non deterministico, relativamente all'occorrenza o non occorrenza di fallimenti, se la sua attivazione o la propagazione dell'errore sono complesse secondo almeno uno dei due modi sopra descritti. Ci si riferisce a tali tipi di guasti con il termine **Mandelbugs**, in riferimento a Benoît Mandelbrot, noto ricercatore nel campo della geometria dei frattali.

Quando invece i guasti si manifestano sotto ben definite condizioni si parla di **Bohr-bugs**, in riferimento al modello atomico di Bohr[18].

Un'altra categoria di bugs, che recentemente sta attirando l'attenzione di molti ricercatori, è rappresentata dagli *aging related bugs*. Molto spesso, i sistemi che sono in esecuzione per lungo tempo, tendono a mostrare prestazioni degradate e un aumento del tasso di occorrenza dei guasti. Se infatti, quest'ultimo fosse costante non influenzerbbe l'occorrenza del fallimento. Tale fenomeno è chiamato **Software Aging**. Si tratta di un fenomeno, osservato empiricamente in numerosi sistemi software, nel quale il sistema degrada con il tempo [5]. In particolare un fallimento può assumere la forma di un servizio totalmente o parzialmente incorretto (es. risultati errati o graduale aumento dei tempi di risposta), hang e/o crash del sistema. Cause tipiche di questo fenomeno sono dovute all'accumulazione di errori, per esempio errori di *round-off*, corruzione dei dati, porzioni di memoria non correttamente deallocate, esaurimento delle

risorse del sistema operativo [6].

Un esempio di errori di round-off accumulati è rappresentato dall'evento capitato il 25 febbraio del 1991, durante la Guerra del Golfo [5], in cui 28 soldati americani morirono e 97 furono feriti, quando il sistema di difesa missilistica Patriot nella caserma di Dhahran, Arabia Saudita, non è riuscito a intercettare un missile Scud in arrivo. La causa del fallimento del sistema missilistico di difesa è stata successivamente associata ad un errore di round-off presente nel software di monitoraggio.

Alcuni autori considerano gli aging related bug come Mandelbugs altri li considerano appartenenti all'intersezione tra Mandelbugs e Bohrbugs. Ad ogni modo, gli aging-related bugs possono essere sia deterministici (per esempio uno o più statement di tipo *delete* mancante) o non deterministici (per esempio un guasto che dipende dall'ordine di arrivo dei messaggi). In entrambi i casi questi tipi di bug richiedono un tempo lungo per manifestarsi (*ritardo rilevante tra l'attivazione dei guasti e l'occorrenza del guasto*). Quindi anche nel caso di bug deterministici probabilmente essi, come i Mandelbugs, si manifesteranno nella fase operazionale del sistema, poiché difficilmente saranno rilevabili in fase di testing. Per questo motivo risulta più corretto considerare gli aging related bug come Mandelbugs[19].

cile ri-produrre tali errori, la loro attivazione risulta in sostanza non deterministica. Gli elusive bug sono stati definiti anche heisenbug, in allusione ai principi di Indeterminazione di Heisenberg. Questa definizione è dovuta alla complessità delle condizioni di manifestazione degli heisenbug che sono tali da scoraggiare ogni tentativo di riproduzione poiché si manifestano raramente e le condizioni di attivazione dipendono da combinazioni complesse dello stato interno del sistema e dell'ambiente esterno. Ad esempio le condizioni di attivazione possono dipendere dall'interazione dell'applicazione e servizi, librerie, virtual machine, middleware e sistema operativo.

1.3 Analisi della letteratura

I requisiti di reliability e availability del software sono in costante crescita, a causa della natura delle odierne applicazioni. Tali requisiti impongono stringenti tempi di indisponibilità di servizio e garanzie di correttezza delle operazioni, in termini di continuità del servizio. Tale necessità nasce dalle possibili conseguenze provocate dai fallimenti del sistema software che possono risolversi in enormi perdite economiche o gravi rischi per vite umane.

Garantire tali requisiti risulta molto difficile per applicazioni particolarmente complesse, sia se progettate da zero che a partire dalla combinazione di componenti software riusabili, i cosiddetti COTS (Commercial Off-the-Shelf component).

1.3.1 Software Rejuvination

Come introdotto in 1.2.1, esiste una particolare categoria di bug che sfuggono a tutte le operazioni di analisi e testing del software. Tali bugs si manifestano con il protrarsi dell'esecuzione delle applicazioni. Durante l'esecuzione del software, tali bugs possono provocare fallimenti transienti che a loro volta possono portare conseguenze non predibili ed eventualmente tradursi in procedure di recupero molto costose.

Inizialmente, le strategie adottate per venire incontro a tali problematiche, sono state di tipo reattivo, e cioè venivano intraprese un certo numero di azioni dopo il verificarsi di un fallimento. Huang et al. in [7] hanno proposto una tecnica chiamata ‘Software Rejuvination’ mediante la quale i) il sistema software viene saltuariamente terminato; ii) vengono eliminate le condizioni di errore accumulate; iii) il sistema viene riportato in uno stato interno corretto.

Con Software Rejuvination si intende un'operazione di rollback periodico e preventivo su applicazioni che sono continuamente in esecuzione, al fine di prevenire fallimenti in futuro.

La schedulazione ottima delle operazioni di Software Rejuvination dipende dalla particolare applicazione ed al centro di numerosi lavori importanti sul piano tecnico scien-

tifico. Esempi di ripristino dello stato corretto sono la garbage collection, il flushing delle tabelle del kernel del sistema operativo, la reinizializzazione delle strutture dati [7, 10, 8], il restarting dei processi dallo stato iniziale o da uno stato relativo all'ultimo checkpoint effettuato.

In letteratura si utilizzano due differenti approcci nello studio del software aging, l'*analytical modeling* e il *measurement based*.

1.3.1.1 Approcci di modellazione analitica

I modelli analitici, tipicamente reti di Petri stocastiche o catene di Markov [9], vengono utilizzati per rappresentare il degrado del sistema. La risoluzione di questi modelli ha l'obiettivo di ottenere la schedulazione ottima per l'attività di Software *Rejuvination*. Huang et. all [7] modellano il sistema mediante catene di Markov tempo continuo. Vengono rappresentati quattro stati:

- Stato 0: highly robust state
- Stato 1: failure probable state
- Stato 2: failure state
- Stato 3: software rejuvenation state.

In tale modello vengono espresse il downtime (tempo in cui il sistema non è disponibile) e i costi (in termini di parametri nel modello) dovuti al downtime, durante le attività di Software Rejuvination.

Un difetto delle tecniche di modellazione analitica consiste nell'accuratezza dei risultati ottenibili per la valutazione delle tecniche di schedulazione per la Software Rejuvination. Esse dipendono fortemente dalla bontà dei modelli, dalle assunzioni fatte e dalla precisione dei parametri utilizzati per risolvere i modelli. In [9] gli autori propongono un modello semimarkoviano basato sulle misure per il workload del sistema. A partire da una definizione di un insieme di stati di workload ottenuta attraverso la cluster

analysis [1], viene stimato il Time To Exstauction (TTE) per ciascuna risorsa e stato di Workload considerato.

1.3.1.2 Approcci basati su misure sperimentalni

L'idea principale di questi approcci è monitorare periodicamente il sistema, collezionare i dati sulle variabili indicative della salute del software in esecuzione e applicare tecniche di rilevazione dei trend statistici per individuare e validare l'esistenza di aging utilizzando la metrica TTE (Time To Exhaustion) [6].

In [6] sono state monitorate 9 Workstations Unix per 53 giorni utilizzando un tool di monitoraggio basato su SNMP (Simple Network Management Protocol), un protocollo applicativo che offre servizi di gestione della rete nella suite di protocolli di Internet. Delle nove macchine monitorate, soltanto tre non hanno riportato fallimenti. Sono stati attentamente analizzati i fallimenti che mostravano esaurimento delle risorse e le serie storiche attraverso il test di Kendall[20] per rilevare presenza di trends. Infine il metodo di Sen [20] è stato utilizzato per stimare la pendenza dei trend rilevati.

1.4 Il ruolo del workload

Tra gli approcci per contrastare o stimare il Software Aging, l'approfondimento di quelli basati sulle misure sperimentalni ha contribuito alla nascita di nuovi scenari nell'ambito della ricerca scientifica e cioè quelli incentrati sul legame tra il fenomeno e il Workload applicato. Un lavoro molto interessante sull'analisi del Software Aging basata sul Workload è [11]. Vengono presentati i risultati di un'analisi condotta sullo stesso insieme di workstation del lavoro precedente, prendendo però in considerazione anche i parametri del Workload applicato. Gli stati di Workload sono identificati mediante la tecnica di analisi dei cluster come anche in [9]. Il modello dello spazio degli stati è costruito determinando le distribuzioni di tempo di soggiorno e definendo una funzione di reward, basata sul tasso di esaurimento delle risorse per ciascuno stato di Workload.

Risolvendo il modello si ottengono i trends relativi all'esaurimento delle risorse e il TTE, per ciascuna risorsa considerata.

I risultati mostrati in [11] confermano che le dinamiche di aging sono relazionate agli stati di Workload e che addirittura *la metodologia workload-driven risulta essere molto più precisa per modellare il fenomeno del software aging*. Tuttavia, gli indicatori di workload utilizzati in suddetti studi rappresentano parametri interni (cioè a livello di sistema) e per questo sono fattori difficilmente controllabili, secondo quanto detto in [13].

Un lavoro indirizzato verso l'impatto di workload a livello applicativo sulle dinamiche di aging è presentato in [15]. In questo caso, però, si tratta di un'applicazione specifica, Apache Web Server, quindi vengono considerati dei parametri di Workload specifici dell'applicazione.

In tutti questi casi, dunque, vengono presi in considerazione parametri di workload specifici del sistema o della particolare applicazione in analisi, non rendendo possibile un confronto tra risultati ottenuti per sistemi diversi [13].

Nel presente lavoro di tesi viene analizzato il legame tra il Software Aging e il Workload. Lo scopo è quello di analizzare, attraverso una caratterizzazione di alto livello del workload, quali fattori del workload indipendenti dall'applicazione, influenzino statisticamente il fenomeno del software aging. Il metodo è basato su un processo di caratterizzazione del workload che, partendo da una descrizione ad alto livello, conduce alla progettazione di un insieme di esperimenti da eseguire su una o più applicazioni da esaminare. Tale caratterizzazione consiste nel determinare un insieme di parametri indicativi del workload, indipendenti dall'applicazione, identificati in intensità del workload, dimensione del workload, tipo e variazione del tipo di richiesta.

1.4.1 Benchmarking

Lo scopo è quello di individuare una metodologia per riuscire ad astrarre' il concetto di workload, in modo da poter applicare le soluzioni trovate a *qualsiasi* tipo di sistema

e renderne possibile la comparazione, in termini di ‘sofferenza di aging’, una sorta di ‘aging benchmarking’. In questo modo è possibile conoscere i comportamenti dei sistemi analizzati, sulla base del Workload ad essi applicato, rendendo possibile la comparazione tra sistemi diversi.

1.4.2 Predittori online

Un altro importante risultato, una volta individuato il legame tra il Workload applicato ed il fenomeno, sarebbe quello di utilizzarlo nella realizzazione di modelli di previsione che stimino la tempificazione per le attività di Software Rejuvenation. Infatti, conoscendo quanto il workload influisce sulle dinamiche di aging, si può stimare il Time To Exaustion sulla base del Workload sottoposto al sistema in un certo periodo e conoscendo quindi come il sistema si comporterà successivamente, evitando così di monitorare continuamente le risorse.



Even more important than learning about statistical techniques is the development of what might be called a capability for statistical thinking.

(Ancora più importante della conoscenza delle tecniche statistiche è lo sviluppo di quello che potremmo chiamare una capacità del pensiero statistico.)

G. E. P. Box, W. G. Hunter e J. S. Hunter

Capitolo 2.

Caratterizzazione del workload

Diversi lavori hanno indagato sulla dipendenza tra il fenomeno del Software Aging e il workload [12, 11, 10, 15, 14], mostrando l'importanza nel considerare il workload nei modelli che controllano le attività di Software Rejuvination. In tali lavori vengono presi in considerazione parametri di workload specifici del sistema o della particolare applicazione in analisi, non rendendo possibile un confronto tra risultati ottenuti per sistemi diversi [13]. Nel presente lavoro di tesi il legame tra il Software Aging e il workload viene analizzato partendo da una caratterizzazione di alto livello del workload, per capire se i fattori così introdotti, indipendenti dall'applicazione, influenzino statisticamente i trend di Software Aging. Si passa successivamente ad una caratterizzazione del workload dipendente dall'applicazione che conduca alla progettazione di un insieme di esperimenti da eseguire su una o più applicazioni da esaminare. Per pianificare tali esperimenti è utilizzata la tecnica del Design Of Experiment, un approccio sistematico per l'indagine di un sistema o un processo. Si osservano dunque le influenze che i fattori, scelti opportunamente, hanno sulle variabili di risposta scelte. Il DOE mira a minimizzare la lista di esperimenti da eseguire al fine di avere risultati statisticamente significativi.

Lo scopo della caratterizzazione del workload è quello di avere un insieme di parametri che descrivano il carico il più possibile in maniera indipendente dalla specifica applicazione. Sulla base di numerosi studi, aventi l'obiettivo di indagare sulla dipendenza del Software Aging dal workload [12, 11, 10, 15, 14], sono stati identificati i requisiti che devono essere soddisfatti per la definizione del workload:

- I parametri descriventi il workload di differenti applicazioni dovrebbero essere *confrontabili*. Dovrebbero, cioè, essere descritti da una prospettiva di alto livello senza riferirsi a parametri specifici dell'applicazione.
- Il comportamento delle applicazioni varia a seconda del workload applicato, sulla base della *quantità* e del *tipo* di lavoro. Una descrizione di alto livello terrà presente soltanto queste due grandezze.

- I parametri di workload indipendenti dalla specifica applicazione devono poter essere *specializzati* in parametri dipendenti dall'applicazione, rendendo possibile l'implementazione sperimentale a partire da suddetta caratterizzazione.
- I valori assunti dai parametri devono essere *realistici*.
- Il numero dei parametri di workload deve rimanere ragionevolmente *basso* per permettere l'esecuzione degli esperimenti in costi contenuti. In particolare tale numero potrebbe tener presente del budget (in termini di tempo e risorse disponibili).

Per definire parametri di workload che rispettino suddetti requisiti si procede ad una *caratterizzazione del workload ad alto livello*, la quale viene poi raffinata in una *caratterizzazione del workload dipendente dalla specifica applicazione* in esame per permettere la messa in pratica di casi di studio reali.

2.1 Caratterizzazione ad alto livello

Si assume che ad alto livello, dal punto di vista dell'utente, il carico sottoposto ad un'applicazione sia rappresentato da una generica richiesta *di servizio*, caratterizzata da:

- *Intensità*. Indicativa del livello di stress sottoposto all'applicazione, misura il numero di richieste nell'unità di tempo sottomesse al sistema.
- *Dimensione*. Rappresenta l'occupazione di memoria complessiva necessaria per i dati scambiati in input e output tra l'utente che richiede il servizio e l'applicazione che lo fornisce.
- *Tipo di richiesta*. E' rappresentativo della diversità delle richieste servite dal sistema, si riferisce ad una classificazione, in seguito descritta, del tipo di operazione che viene effettuata. Questo parametro permette di sollecitare potenziali parti

diverse del sistema specificando il *tipo* di richiesta appartenente ad un particolare insieme.

- *Variazione* del tipo di richiesta. Fissando questo parametro si decide se data una richiesta r la successiva sarà ancora r o un'altra richiesta.

2.2 Progettazione degli esperimenti

Lo scopo principale della ricerca sperimentale è quello di verificare se al variare di certe condizioni (determinate dai cosiddetti fattori sperimentali o trattamenti) i risultati sperimentali (variabili di risposta) variano sistematicamente. La tecnica del Design Of Experiment rappresenta un approccio di tipo sistematico per l'indagine di un sistema o di un processo. Si tratta di una pianificazione di una serie di esperimenti di misurazione nella quale i cambiamenti di una o più variabili di risposta sono causati da uno o più fattori in input al sistema. Il DOE mira a minimizzare la lista di esperimenti da eseguire al fine di avere risultati *statisticamente significativi*. Nell'ambito della ricerca scientifica altri lavori hanno utilizzato questa tecnica [12] [13].

Esempi di applicazione del DoE comprendono:

- valutazione e confronto di configurazioni di progetto
- valutazione di alternative sui materiali da utilizzare
- determinazione dei parametri chiave rispetto all'influenza che essi hanno sulle prestazioni di un sistema

Numerosi casi di ricerca industriale utilizzano questa tecnica, come riportato in tabella 2.1.

2.2.1 Principi base del Design of Experiment

Block what you can, randomize what you cannot.

I principi base su cui si basa il DoE sono:

Tabella 2.1: Esempi di applicazioni industriale del DoE

		esperimenti fisici	esperimenti simulati	esperimenti fisici e simulati
progettazione di prodotto	innovazione		• celle di carico per bilance (2 configurazioni)	• robot rampicante su pali • portellone posteriore auto
	miglioramento		• trasduttori forza per robot	
progettazione di processo	innovazione	• sviluppo software (servizi multimediali Internet)		• misura errori geometrici su CMM • misure microgeometria con profilometro ottico
	miglioramento	• macinazione minerali duri (settore sanitari) • fabbricazione pulegge (auto) • montaggio sedi valvole aspirazione e scarico (auto)	• processo di trafilatura di fili di acciaio • floccaggio elettrostatico (tessile)	• misura di microdurezza su materiali metallici porosi

- *Casualizzazione*: l'ordine di esecuzione deve essere casuale. In questo modo gli errori seguono una distribuzione di tipo normale. Non è permessa intromissione di disturbi legati alla sequenza temporale delle prove.
- *Replicazione*: ogni condizione di prova deve essere ripetuta più volte per consentire la stima dell'errore insito nella risposta. Se si replica in successione (senza casualizzare) ogni volta occorre re-inizializzare i livelli dei fattori in esame (per non andare contro a quanto detto per la casualizzazione).
- *Raggruppamento* (blocking): molti studi si prefiggono di trovare i fattori predominanti per un fenomeno, tuttavia la provenienza dei campioni di prova, come ad esempio le condizioni ambientali in cui si svolgono gli esperimenti, può introdurre un errore di variabilità aggiuntivo; in questi casi si ricorre al blocking delle prove.

2.2.2 La metodologia adottata

Il primo passo nella tecnica del Design of Experiment è *determinare gli obiettivi dell'esperimento*, scegliendo le cosiddette variabili di risposta, quelle il cui comportamento si vuole osservare, e selezionando i *fattori* di processo per lo studio, cioè quelle grandezze che potrebbero *potenzialmente* influire sulle *variabili di risposta*. Un particolare valore del fattore è chiamato *livello*. I fattori si distinguono in *controllabili* e *non controllabili* a seconda se il rispettivo livello può essere scelto dallo sperimentatore, o soltanto osservato.

Successivamente viene definita una lista di esperimenti, chiamati *trattamenti*. Ciascun trattamento è ottenuto assegnando un livello a ciascuno dei fattori controllabili.

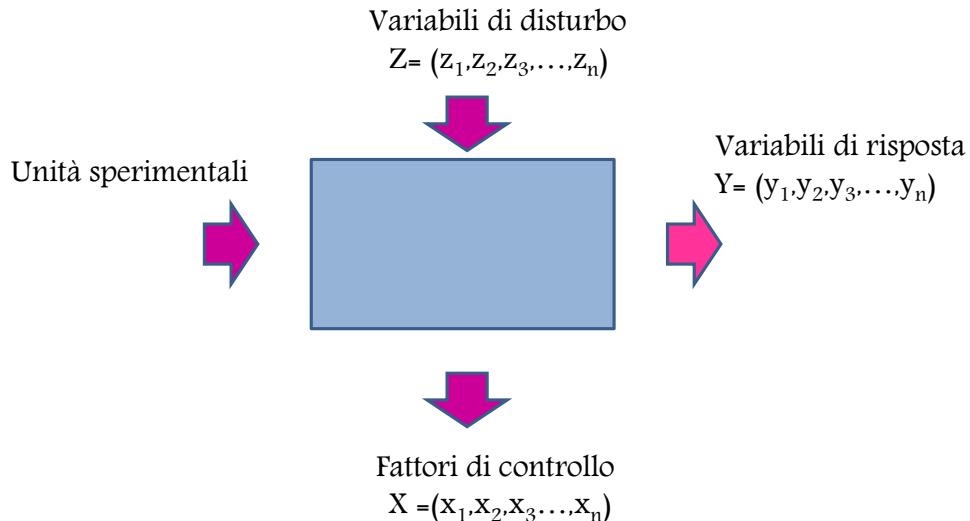


Figura 2.1: Rappresentazione di un sistema sperimentale

Stabilire le modalità di esecuzione di un esperimento richiede conoscenze specifiche in ogni disciplina. Tuttavia, esistono principi generali utili per perseguire questo obiettivo. Quando si utilizzano dati quantitativi, quindi scale di intervalli o di rapporti come nella maggior parte delle ricerche, i *test di significatività* della statistica parametrica sono fondati sul rapporto tra:

- la varianza dovuta ai fattori sperimentali
- la varianza d'errore o non controllata, cioè quella dovuta a fattori non presi in considerazione

Mentre la prima varianza, il numeratore, dipende in buona parte dal fenomeno che si analizza, come la differenza tra gli effetti di due o più farmaci oppure la differenza tra i livelli medi d'inquinamento tra due o più zone, la seconda varianza, il denominatore, essenzialmente dipende dalla *capacità del ricercatore di renderla minima il più possibile*. A tal fine, risulta necessaria un' idonea pianificazione sperimentale, che riesca ad individuare, per il fenomeno oggetto di studio, i *fattori che influenzano la variabile assunta*

a misura del fenomeno, il ruolo che questi fattori hanno nell'analisi e il loro campo di variazione.

Il diverso ruolo dei fattori coinvolti nell'esperimento conduce a scelte strategiche diverse nella pianificazione e quindi a diversi disegni sperimentali. Una primaria distinzione si ha tra *fattore sperimentale o di interesse, fattore sub-sperimentale o fattore blocco, fattore accidentale o di perturbazione.*

Per *fattore sperimentale* si intende quel fattore che si suppone influenzi *direttamente* la variabile di risposta; un esempio classico è dato dal tipo di fertilizzante che influenza la crescita di una particolare specie di pianta. Per **fattore sub-sperimentale** o **di disturbo** si intende generalmente un fattore che può avere effetti sulle variabili di risposta ma non vi è particolare interesse a valutarne l'entità; un esempio di fattore sub-sperimentale o di disturbo è dato dalle condizioni in cui si svolge la prova, come lo specifico giorno di esecuzione degli esperimenti. Nell'esempio citato precedentemente non si considera la tipologia del terreno, esso, infatti, può influenzare *indirettamente* la variabile di risposta agendo *direttamente* sul tipo di fertilizzante. In questo caso la tipologia del terreno costituisce un *fattore sub-sperimentale*. I due disegni appena presentati sono riconducibili rispettivamente al **disegno completamente casualizzato** e al **disegno a blocchi randomizzati**. Se si considerano più fattori, tutti di interesse per lo sperimentatore, si ha il **disegno multifattoriale**. Tali tecniche verranno in seguito approfondite.

2.2.3 Tipologie di DOE

Sulla base di quanti fattori sono presi in considerazione e di come essi si presentano nel piano sperimentale è possibile distinguere diverse tipologie di Design Of Experiment. Le più utilizzate sono le seguenti:

- *Simple Design*
- *Full Factorial Design*
- *Fractional Factorial Design*

La tecnica Simple Design prevede che in ogni trattamento venga variato un solo fattore alla volta per studiare come questi infuisca sulle variabili di risposta. I *limiti* di questa tecnica sono:

- le risorse non vengono ottimizzate;
- potrebbe portare a conclusioni errate in caso di interazione fra i fattori.

Nella seconda tecnica sopra citata viene sperimentata ogni possibile combinazione di ogni livello per ogni fattore. Ad esempio, se per lo studio delle performance di un sistema si sono individuati k fattori, e indichiamo con n_i il numero di livelli del fattore i -esimo, allora il numero totale di trattamenti è:

$$n = \prod_{i=1}^k n_i \quad (2.1)$$

Il notevole *vantaggio* del Full Factorial Design riguarda la possibilità di analizzare oltre agli *effetti primari* dei fattori anche quelli *secondari* e le loro *interazioni*. Il *limite*, invece, è rappresentato dal costo sia in termini di tempo che di risorse impiegate. Ad esempio, un piano sperimentale che prevede 6 fattori con 3 livelli, ciascuno creato utilizzando la tecnica appena descritta, prevede di effettuare $3^6 = 729$ **esperimenti**. Infine la tecnica Fractional Factorial Design è fortemente richiesta quando il numero di fattori o il numero di livelli è elevato. Essa porta con sé i vantaggi del Full Factorial Design, ossia *valutare gli effetti e le interazioni fra fattori*, ed allo stesso tempo *mantiene basso il numero di esperimenti*, dato che viene sperimentata soltanto una frazione degli esperimenti richiesti dal Full Factorial Design. La parte cruciale di questo approccio è rappresentato dal modo con cui viene scelta la frazione. Se i fattori presentano soltanto due livelli di variazione, la campagna di esperimenti può essere facilmente realizzata garantendo le proprietà di *bilanciamento* ed *ortogonalità*. In particolare, per *bilanciamento* si intende che tutte le combinazioni dei trattamenti devono avere lo stesso numero di osservazioni. In generale, due vettori qualsiasi della stessa lunghezza si dicono *ortogonali* se la somma dei prodotti degli elementi corrispondenti è nulla. In

ogni campagna sperimentale, si associa ad ogni fattore un vettore di lunghezza pari al numero di trattamenti. Tale vettore avrà come elementi i valori che il fattore assume nei trattamenti. Se per ogni coppia di vettori, associati alla campagna, il prodotto scalare è nullo allora si dice che la campagna di esperimenti è *ortogonale*. Per poter effettuare i trattamenti in maniera rigorosa, rispettando il principio di *casualizzazione*, bisogna creare un piano sperimentale che preveda:

- assegnazione *casuale* dei livelli dei fattori;
- assegnazione *casuale* dell'ordine delle prove.

2.2.4 Classificazione del DOE in base ai fattori

I disegni sperimentali possono essere classificati sulla base del numero di fattori sub-sperimentali che vengono presi in considerazione. Quelli più frequentemente utilizzati sono:

- *disegno completamente casualizzato*, quando non è tenuto in considerazione alcun fattore; sub-sperimentale, ma solo i fattori sperimentali e i fattori casuali;
- *disegno a blocchi randomizzati*, nel caso in cui sia presente un solo fattore sub-sperimentale;
- *disegno multifattoriale*, nel caso in cui siano presenti due o più fattori sub-sperimentali (es. quadrato latino e i quadrati greco-latini).

Tra questi ultimi rientrano anche i disegni fattoriali, nei quali l'attenzione del ricercatore è posta soprattutto sull'analisi delle interazioni tra i due o più fattori presi in considerazione, senza distinzioni tra fattori sperimentali e sub-sperimentali.

2.2.4.1 Disegno completamente casualizzato

Si parla di disegno completamente casualizzato quando l'assegnazione dei trattamenti alle unità sperimentali omogenee è casuale. Si tratta della tecnica più semplice, nella

quale però non viene dato un peso al fattore sub-sperimentale (o di rumore). Essa necessita, inoltre, che le unità sperimentali siano omogenee.

I *vantaggi* più evidenti di questo disegno sperimentale sono:

- facilità dell'esecuzione;
- semplicità dell'analisi statistica;
- varianza d'errore con il numero massimo di gradi di libertà;
- gruppi non bilanciati, fenomeno frequente quando l'osservazione si prolunga nel tempo.

Gli *svantaggi* principali sono:

- raramente in natura si dispone di unità sperimentali omogenee;
- non si tiene presente l'influenza dei fattori di rumore che nella realtà sono quasi *inevitabili*;
- è ugualmente interessante valutare se, per la variabile analizzata, esistono differenze significative anche entro altri fattori, quali il ceppo, l'età, il peso, il sesso e in generale tra i livelli di tutte le variabili ritenute influenti, anche se ovvie.

2.2.4.2 Disegno casualizzato a blocchi

E' detto casualizzato a blocchi un piano in cui le unità sperimentali sono divise in gruppi omogenei, detti blocchi e la successione dei trattamenti in ciascun blocco è casualizzata. Si tratta del caso in cui negli esperimenti la variabilità dovuta a fattori di rumore può influenzare sensibilmente i risultati. I fattori di rumore possono essere:

- *non noti e non controllabili*: non si conosce l'esistenza di un particolare fattore di rumore e la sua influenza sulle variabili di risposta. In tal caso come contromisura si adotta la *casualizzazione* delle prove;

- *noti ma non controllabili*: si riesce a conoscere i valori assunti dalle variabili di rumore ma non si possono controllare tali valori. Si rimedia a questa situazione con l'*analisi della covarianza*;
- *noti e controllabili*: sono noti i valori assunti dai fattori di rumore ed inoltre, cosa più importante, tali valori possono essere *controllati* e quindi impostati secondo le necessità. Una tecnica di progettazione molto diffusa in ambito industriale, chiamata *blocking*, viene utilizzata per eliminare gli effetti di tali fattori nelle tecniche di comparazione statistica tra diversi trattamenti. Si tratta di una pianificazione più specifica e selettiva nella quale l'assegnazione casuale dei trattamenti è fatta entro ciascun blocco. Il modello sottostante al disegno a blocchi randomizzati è:

$$y_{ij} = \mu + \alpha_i + b_j + e_{ij} \quad j = 1, 2, \dots, n, \quad i = 1, 2, \dots, n \quad (2.2)$$

dove α_i indica il generico effetto i -esimo del fattore sperimentale, b_j indica il generico effetto j -esimo del fattore sub-sperimentale o fattore di blocco.

2.2.4.3 Disegno multifattoriale

Si tratta di un disegno in cui sono presenti più fattori di disturbo. Al fine di evitare l'influenza sistematica di tali fattori bisogna approntare un piano sperimentale che prevenga da eventuali cause di errore di valutazione. La strategia è quella di adottare un piano quadrato in modo da consentire che tutti i trattamenti siano presenti non solo in ciascun gruppo identificato dal blocco B_j ma anche in ciascuna delle posizioni ordinali.

2.2.5 Determinazione del piano sperimentale

In sintesi, esaminati i requisiti che un disegno sperimentale deve soddisfare e analizzate le diverse tecniche di Design Of Experiment, nel determinare un piano sperimentale è fondamentale:

- individuare, per il fenomeno oggetto di studio, i *fattori che influenzano la variabile assunta a misura del fenomeno*
- individuare il ruolo che questi fattori hanno nell'analisi
- individuare il campo di variazione di tali fattori

Con lo scopo di valutare l'influenza del workload sul Software Aging, vengono pianificati una serie di esperimenti variando i valori dei parametri del workload e stimando, quindi, gli effetti dei cambiamenti pianificati sui trend di aging. Le *variabili di risposta* da osservare sono rappresentate dagli indicatori di aging scelti, mentre i parametri di workload identificano i *fattori* sperimentali. Come *variabili di risposta* sono state scelte un insieme di variabili indicative dello stato della memoria del sistema tra cui memory depletion e un insieme di variabili indicative delle prestazioni del sistema tra cui il throughput loss. I parametri di workload di alto livello riportati in 2.1, rappresentano i *fattori* del piano sperimentale e cioè:

- Intensità
- Dimensione
- Tipo
- Variazione

I livelli dei singoli fattori vengono scelti sulla base degli effetti che si vogliono osservare, in particolare in questo lavoro vengono considerati due livelli, rispettivamente il livello *low* e il livello *high*, per ciascun fattore. Per l'*intensità*, il livello low sarà quello relativo ad un basso tasso di richieste da sottoporre al sistema mentre nel caso high il sistema sarà sottoposto ad uno stress elevato, in termini di quantità di richieste nell'unità di tempo. Con *dimensione* low ci si riferirà ad una bassa occupazione di memoria dei dati scambiati tra l'utente e l'applicazione; dimensione high l'opposto. Per *tipo di richieste* light si intenderà un insieme di richieste che sovraccaricano meno il sistema rispetto le richieste di tipo heavy. Verrà successivamente descritto su che base verrà

effettuata questo tipo di classificazione. Infine, *variazione* prevede due livelli, uno relativo all'assenza e l'altro alla presenza di variazione del tipo di richiesta sottoposta ogni volta al sistema. In particolare, nel caso di variazione ogni successiva richiesta verrà scelta in maniera random tra un insieme di richieste possibili.

Un modo pratico per determinare i valori di tali livelli è considerare dati sul campo riguardanti l'applicazione sotto esame. Questo viene realizzato mediante opportuni test, i *preliminary test*, di cui al seguito.

I risultati ottenuti da tali test daranno un'indicazione sui *valori* da attribuire ai *fattori*. Siccome l'approccio utilizzato in questa tesi si presta ad essere utilizzato su qualsiasi tipo di sistema, in questa fase si è data particolare attenzione al fattore *software*. Lo studio si concentra sul comprendere come sia opportuno considerare il fattore software e cioè sulla scelta di quale tipo di progettazione sperimentale utilizzare tra quella casualizzata completamente, a blocchi o quella multifattoriale. Tale necessità nasce dalla considerazione, fatta precedentemente, che il diverso ruolo dei fattori coinvolti nell'esperimento conduce a scelte strategiche diverse nella pianificazione e cioè a diversi disegni sperimentali e quindi scegliere erroneamente di adottare un disegno piuttosto che un altro, potrebbe portare a risultati *non rappresentativi* del fenomeno. Il software rappresenta una ‘condizione in cui si svolge’ l'esperimento e rappresenta quindi un fattore sub-sperimentale (o di disturbo). Il software, infatti, è un fattore che non influenza direttamente le variabili di risposta, ma i *fattori sperimentali scelti*, cioè intensità, dimensione, variazione e tipo di richiesta che a loro volta *influenzano direttamente* le *variabili di risposta sperimentali*. Per eliminare gli effetti di questo fattore, la tecnica di DOE utilizzata è la tecnica randomizzata a blocchi, in cui quindi il software rappresenta un *fattore di blocco* dell'esperimento.

Il piano sperimentale è realizzato mediante il tool JMP [21]. In tale piano vengono analizzati diversi tipi di software. L'output fornito dal tool è caratterizzato da una lista di esperimenti da eseguire mostrata in 4.2, in cui ogni fattore (fattore di blocco: software, fattori sperimentali: intensità, dimensione, tipo di richiesta) prevede sue livelli.

Nome Blocco	Intensità	Dimensione	Tipo di richiesta	Variazione
Software tipo 1	Low	High	Light	YES
Software tipo 1	High	High	Light	NO
Software tipo 1	Low	Low	Heavy	NO
Software tipo 1	High	Low	Light	YES
Software tipo 2	High	High	Heavy	YES
Software tipo 2	Low	High	Heavy	YES
Software tipo 2	Low	Low	Light	NO
Software tipo 2	High	Low	Light	NO
Software tipo 3	Low	Low	Light	YES
Software tipo 3	High	Low	Light	YES
Software tipo 3	High	High	Heavy	NO
Software tipo 3	Low	High	Heavy	NO
Software tipo 4	Low	High	Heavy	YES
Software tipo 4	High	Low	Light	NO
Software tipo 4	Low	Low	Light	NO
Software tipo 4	High	High	Heavy	YES

Tabella 2.2: Piano sperimentale

2.3 Caratterizzazione dipendente dall'applicazione

Poichè lo scopo di questo lavoro è condurre esperimenti reali, nasce l'esigenza di specializzare questi parametri di workload in quelli dipendenti dall'applicazione in analisi. Viene così delineato il preciso significato di tali parametri. L'*intensità* è vista come il numero di richieste di servizio nell'unità di tempo. A questo punto va quindi ben definito il significato di 'richiesta' per l'applicazione che si sta considerando. Lo stesso vale anche per gli altri parametri, e quindi con *tipo di richiesta* ci si riferirà agli insiemi di richiesta che più frequentemente vengono sottoposti al sistema, di seguito descritti. *Dimensione* indicherà la dimensione occupata in memoria dai dati scambiati tra l'utente e l'applicazione (sia in input che in output) e infine *variazione* di richiesta determinerà il tipo di richiesta che verrà eseguito.

2.3.1 Analisi dei cluster

L'analisi dei cluster è un metodo molto utile per creare gruppi o cluster di oggetti in modo che tutti gli oggetti presenti in un certo cluster siano molto simili e oggetti in differenti cluster siano abbastanza distinti. Si tratta di un componente molto impor-

tante nel *data mining*, il processo di esplorazione e analisi di grandi quantità di dati al fine di scoprire informazioni utili. La tecnica del clustering è molto importante anche nel campo della *pattern recognition* che consiste nell'analisi e identificazione di pattern all'interno di dati grezzi al fine di identificarne la classificazione. In letteratura esistono diverse interpretazioni sulle proprietà che devono essere soddisfatte dai dati in input per stabilire la loro appartenenza ad un certo gruppo. Per esempio per Bock (1989) tutti gli oggetti in un cluster devono:

- condividere le stesse o strettamente correlate proprietà;
- mostrare piccole distanze mutue o dissimilarità;
- avere 'contatti' o 'relazioni' con almeno un altro oggetto nel gruppo;
- essere chiaramente distinguibili dai complementi (cioè il resto degli oggetti nell'insieme dei dati).

Carmichael et al. nel 1968 suggerì che gli insiemi contengono clusters di punti se le distribuzioni di punti soddisfano le seguenti condizioni:

- Ci sono regioni dello spazio continue e densamente popolate.
- Tali regioni sono circondate da regioni dello spazio continue e relativamente vuote.

In questo lavoro di tesi si è fatto uso dell'analisi dei cluster per ottenere una classificazione dei tipi di richieste e cioè richieste di tipo *light* e richieste di tipo *heavy*. Vengono sottoposti all'algoritmo i dati ottenuti da esecuzioni sperimentali. Ulteriori approfondimenti sono rimandati al capitolo 4.

2.3.1.1 Preprocessing dei dati

Affinché l'algoritmo di clustering fornisca risultati affidabili sono necessarie una serie di operazioni preliminari.

Data standardization La standardizzazione dei dati rende i dati adimensionali.

Questo tipo di operazione è necessaria quando esistono variabili con unità di misura diverse, poiché in questo caso non risulta corretto effettuare confronti tra i valori delle variabili.

Data correlation E' di fondamentale importanza esaminare le correlazioni a coppie (anche detta *cross-correlazione*) tra le variabili in esame per avere un'idea di quali variabili risultano molto correlate tra di loro e cioè quali variabili fornirebbero informazioni duplicate o sovrapposte. Utilizzare due variabili che sono correlate con un valore di cross-correlazione $> |0,7|$ in un'analisi di regressione o dei cluster potrebbe avere un impatto negativo sui risultati.

Algoritmo k-means L'algoritmo k-means è uno tra i più usati algoritmi di clustering. Esso è progettato per raggruppare dati numerici nei quali ciascun cluster ha un centroide chiamato media. Ad ogni iterazione dell'algoritmo esso procede, dato un numero iniziale di cluster, allocando ciascun dato dell'insieme iniziale al cluster più vicino cambiando l'appartenenza ad un certo cluster sulla base della funzione di errore. Questo procedimento viene effettuato fin quando non cambia più significativamente la funzione di errore o l'appartenenza di un campione ad un certo cluster.

Dato D , un insieme di dati con n istanze e sia C_1, C_2, \dots, C_k i cluster disgiunti di D . La funzione di errore è definita come:

$$E = \sum \sum d(x, \mu(C_i)) \quad (2.3)$$

dove $\mu(C_i)$ è il centroide del cluster C_i e $d(x, \mu(C_i))$ denota la distanza tra x e $\mu(C_i)$ per la quale tipicamente si fa riferimento alla distanza euclidea d_{euc} . L'algoritmo è costituito dalle seguenti fasi:

Dati in input:

Insieme di dati D , numero di cluster k , dimensione d :

C_i l' i -esimo cluster

1. Fase di inizializzazione

1: (C_1, C_2, \dots, C_k) = Initial partition of D .

2. Fase iterativa

2: **repeat**

3. d_{ij} = distanza tra l'oggetto i e il cluster j ;

4. $n_i = \arg \min_{1 \leq j \leq k} d_{ij}$;

5. Assegna l'oggetto i al cluster n_i

6. Calcola i valori medi dei cluster che hanno subito cambiamenti;

7. **until** Nessun ulteriore cambiamento di appartenenza ai cluster avviene nell'iterazione completa;

8. Mostra i risultati.

L'algoritmo appena descritto sarà utilizzato in questa tesi per determinare le classi di appartenenza di tutte le richieste che verranno prese in considerazione.

2.3.2 Test preliminari

All'inizio di questo capitolo è stata presentata la caratterizzazione del workload di alto livello. Prima di eseguire gli esperimenti risulta necessario elaborare alcuni test al fine di determinare i limiti delle applicazioni in esame, assegnando quindi dei valori ai parametri precedentemente citati. L'intensità è rappresentata da una percentuale della capacità massima, in termini di richieste nell'unità di tempo (richieste al secondo) che il sistema in esame riesce a sostenere. Tale limite viene determinato dallo sperimentatore attraverso un *capacity test*. Un altro aspetto importante è rappresentato dalla durata dell'esperimento. Questa dovrebbe essere pari ad un valore sufficiente ad osservare un significativo trend di aging nei dati sperimentali. La stima di tale valore si ottiene dal *test zero*.

2.3.2.1 Capacity test

Un capacity test serve a determinare la capacità massima che un sistema è in grado di sostenere, in modo da dimensionare il carico in fase sperimentale. La procedura adottata in questo test è la seguente:

- determinare una metrica per misurare il throughput del sistema;
- sollecitare il sistema con un carico di intensità sempre crescente, fino a quando nel grafico del throughput si osserva un ‘ginocchio’cioè una stabilizzazione del numero di richieste soddisfatte, nell’unità di tempo.

Il ‘ginocchio’ indica il limite della capacità massima del sistema. Oltre tale limite il sistema non è più in grado di servire le richieste opportunamente (a partire dal ginocchio all’aumentare del numero di richieste il throughput non aumenta ma rimane lo stesso o addirittura diminuisce). Siccome, data un’applicazione, i limiti possono essere raggiunti con differenti tipi di richieste, i capacity test devono essere eseguiti per ciascun tipo di richiesta. Ad ognuna di queste corrisponderà, dunque, un certo limite, inteso come il numero massimo di richieste di quel tipo che il sistema riesce a soddisfare, nell’unità di tempo. Quest’ultima scelta assicura che durante gli esperimenti, il sistema non fallisca a causa del superamento della capacità del sistema (*sovraffatto del sistema*). In questo modo è possibile separare i fallimenti dovuti all’aging del software da quelli causati dal sovraffatto del sistema.

2.3.2.2 Test zero

Ciascun esperimento dovrebbe durare un tempo sufficiente a osservare un trend significativo nei risultati. La durata degli esperimenti è una grandezza dipendente dal sistema. Se definissimo a priori la durata degli esperimenti per tutte le applicazioni, potremmo eseguire un certo numero di esperimenti inutilmente, se la durata T fosse troppo bassa, a causa del numero insufficiente di campioni su cui osservare un trend di aging, o dover eseguire esperimenti di costo elevato, se T fosse troppo alto. Assumendo

che con parametri di workload meno intensi si osservano trend di aging meno marcati, il test zero ha lo scopo di individuare se è osservabile un trend di aging con il carico meno stressante, sottoposto al sistema. Questo test infatti viene eseguito settando i parametri di workload al loro minimo livello. A questo punto la durata dell'esperimento per ciascun sistema è valutato attraverso un algoritmo sviluppato e presentato in [22]. Tale algoritmo determina il tempo minimo al fine di osservare trends statisticamente significativi, fissata una certa variabile di risposta. Dato un livello di confidenza in input esso stima se il numero di campioni collezionati fino ad un tempo t è sufficiente per ottenere un trend significativo.

2.4 Rilevazione e stima dei trend statistici

In numerosi problemi nell'ambito scientifico un importante obiettivo è quello di rilevare e stimare le dinamiche dei parametri monitorati al fine di capire se esistono delle dinamiche particolari nel fenomeno che si vuole descrivere. Le figure 2.2 mostrano un esempio di assenza di trend, presenza di trend casuale e infine trend non casuale.

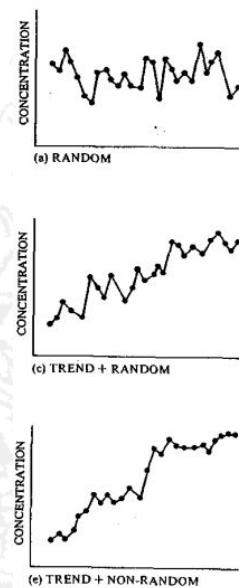


Figura 2.2: Esempi di trend statistici

2.4.1 Il test Mann Kendall

Il test Mann-Kendall [20] è un test non parametrico per la regressione lineare di dati valutati nell'evoluzione temporale. L'obiettivo principale di questo test è valutare l'ipotesi nulla H_0 relativa all'assenza di trend, contro l'ipotesi alternativa H_1 relativa alla presenza di trend (ascendente o discendente). In tale test viene elaborata la statistica S_i di Mann-Kendall:

$$E = \sum_{k=1}^{n-1} \sum_{l=k+1}^n sgn(y_l - y_k) \quad (2.4)$$

Con $l > k$ e $sgn(x)$ la funzione:

$$sgn(x) = \begin{cases} -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases} \quad (2.5)$$

dove n è il numero di punti dei dati e y_l è il dato per l' l -esimo punto.

Per testare l'ipotesi nulla, H_0 è rigettata in favore dell'ipotesi H_1 di trend ascendente (discendente) se S è positivo (negativo) nel caso in cui il valore critico corrispondente al valore assoluto di S è minore rispetto al livello di significatività del test, α .

Il test di Mann-Kendall è semplice, efficiente e robusto rispetto eventuali dati mancanti. Inoltre, non è necessario che i dati seguano una particolare distribuzione statistica, perciò si tratta di un test non parametrico.

2.4.2 La stima della pendenza di Sen

Una volta confermata la presenza di trend, può essere valutata la sua pendenza calcolandone la stima dei minimi quadrati, attraverso i metodi di regressione lineare, facendo particolare attenzione a rimuovere gravi errori e outliers nei dati che potrebbero portare

a risultati totalmente diversi. Per superare questa limitazione, *Sen* ha sviluppato una procedura non parametrica. Si tratta di un metodo che non viene molto influenzato dalla presenza di outlier ed anche molto robusto rispetto alla mancanza di dati.

N' pendenze sono calcolate per tutte le coppie di punti con l e k in cui $l > k$ come:

$$(y_l - y_k)/(l - k) \quad (2.6)$$

Queste pendenze sono poi classificate e ne viene calcolata la mediana. Quest'ultima è richiesta per la stima della pendenza.



L'immaginazione è più
importante della conoscenza

Albert Einstein

Capitolo 3.

Studio di un caso

Questa tesi ha lo scopo di studiare il legame tra il Software Aging e il workload sottoposto ad un sistema software. Per fare questo si è scelto di implementare un caso di studio realistico. La scelta del caso di studio deriva da un attento studio delle problematiche riportate in alcuni sistemi software. Il DBMS MySQL, molto utilizzato in ambito scientifico, ha riportato diverse anomalie nel corso degli anni. Verrà presentata una specificazione del workload di MySQL, le scelte fatte sulle modalità di implementazione di tale caso di studio. In particolare si è scelto di utilizzare un benchmark per DBMS OLTP. Infine, verranno presentati i valori scelti per ciascun livello del piano sperimentale adottato.

3.1 MySQL

MySQL, il più popolare Database Management System relazionale Open Source, è sviluppato, distribuito e supportato da Oracle Corporation. MySQL è molto veloce, affidabile e facile da usare. L'interesse di sperimentare questo sistema software nasce dallo studio di diversi bug reports verificati che mostrano numerosi bachi gravi in alcune versioni del software. Tali bachi software riguardano più di un sottosistema di Mysql, in particolare InnoDB, Replication e Optimizer e sono sia relativi alla memoria che ad altre risorse. Alcuni errori rilevati riguardano casi di memory leak, variabili non inizializzate, crash del sistema, corruzioni dei dati, out of memory. E' evidente che la presenza di tali reports non implica necessariamente che il software Mysql sia affetto da aging, dimostrare che lo sia rappresenta uno degli obiettivi di questa tesi.

3.1.1 Specificazione del Workload

Questa fase consiste nel caratterizzare il workload del DBMS MySQL secondo i parametri descritti nel 2:

- Intensità
- Dimensione
- Tipo di richiesta
- Variazione

Un primo passo fatto per raggiungere quest'obiettivo è stato quello di studiare lavori presenti in letteratura per capire se ci fosse già una classificazione del workload di MySQL di questo tipo. La maggior parte dei lavori presenta degli studi sulla composizione degli statement sql e sulla percentuale con cui essi sono presenti nelle applicazioni. Altri studi si incentrano su workload forniti dai benchmark o su workload classificati in base alle classi di query. Per classificare il workload sulla base dei parametri sopracitati, si è pensato di considerare le query come elemento basilare della classificazione, riferendosi a [23] o anche [24] nel quale il workload è caratterizzato dalle query.

In quest'ottica il workload specifico di Mysql sarà costituito da:

- *Intensità.* Numero di query eseguite al secondo
- *Dimensione.* Numero di tuple moltiplicato per la dimensione della tupla restituita e/o modificata dalle query (Per tupla si intende un generico elemento di una relazione con attributi in un database relazionale.)
- *Tipo di richiesta.* Sulla base del tipo di operazione da effettuare, le richieste vengono suddivise in quattro macrogruppi:
 - visualizzazione (selezione, proiezione,...)
 - visualizzazioni innestate
 - visualizzazione con congiunzione (unione)

- modifica (aggiornamenti, inserimenti, cancellazioni,...)

In particolare, sulla base di un'algoritmo di cluster, di cui al capitolo 2, tali richieste verranno ulteriormente suddivise in due gruppi, richieste di tipo light, richieste di tipo heavy. Poiché l'analisi del cluster può essere fatta se sono disponibili i campioni su cui effettuare il clustering, si rimanda questo tipo di caratterizzazione al capitolo 4.

- *Variazione*

Si è pensato di prevedere due possibili tipi di variazione:

- *Assenza di variazione*. Fissato un certo tipo di richiesta (light o heavy), per tutta la durata dell'esperimento verrà eseguita *sempre* quel tipo di richiesta. Per ciascuna richiesta appartenente ad un certo tipo, sono previste più query (esempio selezione prevederà 3 tipi di selezione riguardanti campi e tavole diversi, al fine di ‘sollecitare’ il sistema in modo più realistico possibile).
- *Presenza di variazione*. Fissato un certo tipo di richiesta, per tutta la durata dell'esperimento vengono eseguite, in maniera random, i sottogruppi di richieste appartenenti a quel determinato tipo. Risulta quindi *fondamentale* stabilire quali siano le richieste di tipo *light* e quali quelle di tipo *heavy*.

3.1.2 Benchmark di DBMS

Per implementare un caso di studio esistono diverse possibilità:

- realizzare un'applicazione ex-novo;
- utilizzare applicazioni già esistenti.

La scelta di realizzare un' applicazione ex-novo non è sempre la più giusta. In tal caso, infatti, c'è sempre la possibilità che si verifichino errori di progettazione o comunque che il progettista venga messo davanti al difficile compito di implementare tale applicazione nella maniera giusta. La scelta opportuna da fare è quella di valutare in ciascuna

situazione quale sia il trade-off tra costi e benefici ottenuti. Infatti, nel caso in cui esista già un'applicazione che implementi le operazioni necessarie al caso di studio, e non esistono controindicazioni, la creazione di un'applicazione ex-novo risulta infruttuosa. Per queste importanti considerazioni si è scelto di far riferimento ai cosiddetti benchmark. Si tratta di *carichi pilota* utilizzati principalmente per:

- confronto delle prestazioni per sistemi diversi per una loro *selezione*;
- valutazione di un sistema in esercizio per la sua *ottimizzazione*;
- proiezioni del carico attuale di un sistema in esercizio per scopi *previsionali*.

I benchmark si dividono in standard e personalizzati (di utente). Nel primo caso si adotta una metodologia fissa, per qualsiasi tipo di applicazione o ambiente. Essendo collaudati da molte applicazioni, tali tipi di benchmark risultano i più robusti. Nel secondo caso, invece, i benchmark si adattano allo specifico ambiente, in modo da ottenere una rappresentazione ‘fedele’ del sistema sotto studio, scegliendo opportunamente i parametri di caratterizzazione. In generale si può affermare che in fase di acquisizione di un sistema è utile utilizzare i benchmark standard per una prima selezione, e in una fase successiva, per il confronto tra due o tre sistemi, i benchmark di utenti.

Dunque, nell’ambito della valutazione delle prestazioni di un sistema, risulta fondamentale l’utilizzo di un benchmark.

3.1.2.1 Motivazioni

Questo lavoro non ha come scopo quello di valutare le prestazioni del DBMS MySQL, ma di sperimentare se esso soffre di aging ed in caso affermativo, quali potrebbero essere i parametri di workload che maggiormente influenzano tali dinamiche. A tal fine, i benchmark standard possono essere utilizzati per implementare un caso di studio secondo regole ben definite.

I benchmark per DBMS, infatti, danno l’opportunità di:

- generare un database;

- popolare il database con dati forniti dal benchmark;
- disporre di query e/o transazioni forniti dal benchmark;
- fornire al database un'opportuna struttura in termini di chiavi e indici.

Esistono numerosi benchmark per DBMS in commercio. Tali benchmark si suddividono sulla base del tipo di applicazione da testare. In particolare quelle di tipo Online Transaction Processing (**OLTP**) e On Line Analytical Processing (**OLAP**).

OLTP rappresenta la tradizionale elaborazione di transazioni adottata dai processi operativi, in cui ogni operazione (predefinita e relativamente semplice) coinvolge un numero ridotto di dati. Tali dati devono soddisfare le proprietà **ACIDe** delle transazioni (Atomicità, Coerenza, Isolamento, Durabilità).

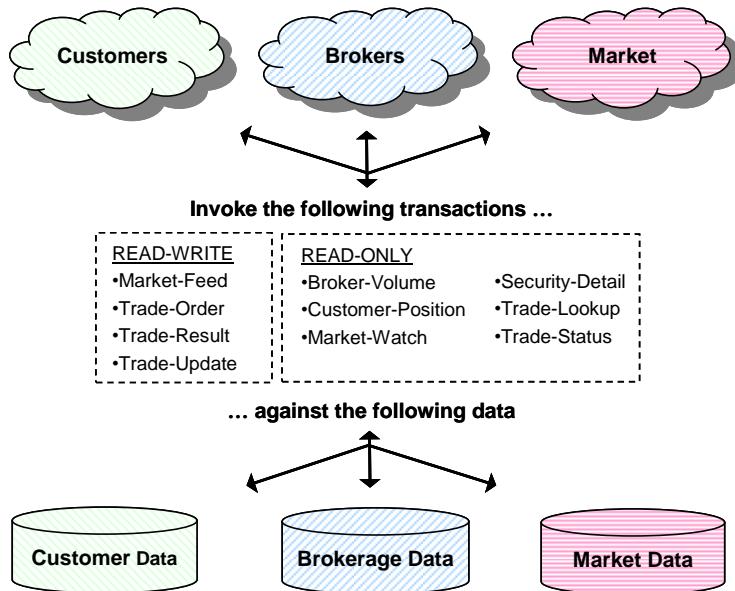
OLAP rappresenta l'elaborazione di operazioni per il supporto alle decisioni, in cui ogni operazione (complessa e casuale) può coinvolgere molti dati, storici. Le proprietà **ACIDe** risultano non rilevanti, tipicamente tale tipo di elaborazione è utilizzato per operazioni di lettura.

In questo lavoro si fa riferimento ad un database di tipo OLTP, costituito da numerose operazioni con una limitata complessità computazionale e una durata limitata. Questa scelta deriva dalla considerazione che si vuole 'stressar' il sistema MySQL sottoponendogli quante più operazioni, quindi relativamente semplici. Tali operazioni daranno sicuramente dei risultati significativi in termini di memoria utilizzata e produttività.

3.1.2.2 Il benchmark TPC-E

Uno tra i benchmark per DBMS di tipo OLTP molto conosciuti è il benchmark TPC-E. In particolare, tale benchmark simula il carico di una società di mediazione. Vengono sottoposte al sistema in esame un insieme di transazioni intense di sola lettura e/o aggiornamento, le quali emulano le attività previste in ambienti di applicazioni OLTP complessi. Lo schema del database, la popolazione dei dati, la composizione delle transazioni e le norme di attuazione sono stati progettati per essere ampiamente rappresentativi dei moderni sistemi OLTP. Nello specifico, TPCE modella l'attività di

un'azienda di mediazione, la quale deve gestire conti di clienti, eseguire ordini commerciali e occuparsi dell'interazione tra il cliente e i mercati finanziari. Il diagramma 3.1.2.2 illustra il flusso delle transazioni del modello di affari e i componenti chiave contemplati nel benchmark.



Come detto in precedenza, lo scopo del benchmark, è quello di sottoporre un *carico pilota* ad un particolare sistema, per valutarne le prestazioni. A tal proposito, le transazioni adottate dal benchmark TPC-E soddisfano particolari regole, come quelle relative al flusso di esecuzione delle transazioni stesse, in modo tale che il modello di carico sia quanto più possibile *rappresentativo* di un sistema produttivo *reale*. Poiché, come sopra citato, lo scopo di questa tesi non è valutare le prestazioni del DBMS MySQL ma studiare e valutare il fenomeno di aging in suddetto sistema, il benchmark TPC-E è stato utilizzato per poter implementare un caso di studio secondo **regole ben definite e replicabili**.

3.1.2.3 Struttura database del benchmark

Il database del TPC-E è costituito da 33 tabelle organizzate in quattro macroinsiemi:

- Customer: 9 tabelle che contengono informazioni relative ai clienti dell'azienda di mediazione.

- Broker: 9 tabelle relative all'azienda e i dati di mediazione.
- Market: 11 tabelle contenenti informazioni su aziende, mercati, scambi e settori industriali.
- Dimension: 4 tabelle di dimensione che contengono informazioni come indirizzi e codici postali.

3.1.2.4 Passi per l'utilizzo del benchmark

Secondo tali premesse, il benchmark TPC-E è stato utilizzato seguendo i passi:

- **Creazione database.** In seguito allo studio del benchmark TPC-E si è passati a reperire dal sito dello standard il tool per la creazione del database ed esecuzione del benchmark. Mediante degli script sql viene creato il database e ciascuna tabella prevista dal benchmark.
- **Popolamento del database.** Attraverso degli script sql vengono caricati tutti i dati nel database precedentemente creato.
- **Creazione indici, chiavi esterne del database.** In ultimo, vengono create le opportune chiavi esterne e gli indici previsti dal benchmark.

Nel contesto dei database relazionali, si definisce[25]:

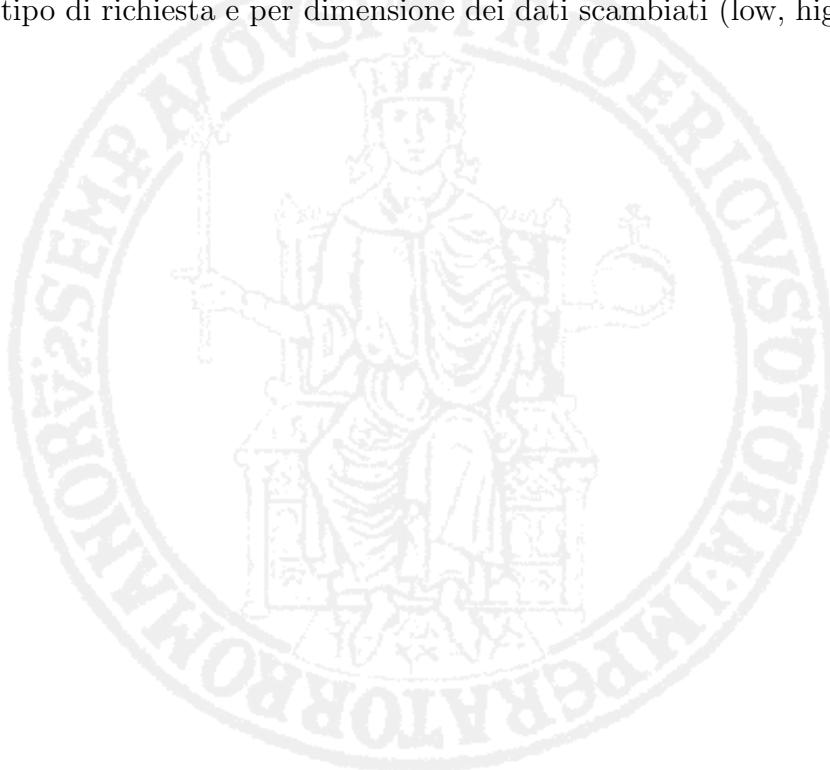
- Superchiave di una relazione r : un insieme di K attributi tali che r non contiene due tuple distinte t_1 e t_2 con $t_1[K] = t_2[K]$.
- Chiave di una relazione r : una superchiave K minimale di r (non esiste un'altra superchiave K' di r che sia contenuta in K come sottoinsieme proprio)

Un indice è una struttura dati realizzata per migliorare i tempi di ricerca dei dati.

L'appendice A contiene struttura del database TPC-E utilizzato.

3.1.2.5 Le Query TPC-E considerate

Il benchmark TPC-E consiste di 11 transazioni che coprono un'enorme varietà di funzioni del sistema. Nell'usuale elaborazione del benchmark, esse vengono eseguite in un ordine tale da generare il workload desiderato mantenendo comunque l'ambiente di esecuzione *semplice, ripetibile e facile da eseguire*. Nel contesto dell'utilizzo di tale benchmark per questo lavoro di tesi, non è stato necessario utilizzare transazioni così fatte ma è bastato prelevare da esse le rispettive query. In una prima fase è stato selezionato un sottoinsieme di query dall'insieme di quelle rese disponibili dal benchmark. La scelta è stata fatta sulla base delle tabelle a cui le transazioni si riferiscono e della classificazione del workload specifico per MySQL menzionata in precedenza. Inoltre tali query sono state classificate anche sulla base delle dimensioni dei dati ritornati, secondo sempre suddetta classificazione. Risulta ovvio che diverse dimensioni dei dati scambiati vengono ottenuti con query diverse. L'appendice B mostra le query TPC-E utilizzate nel contesto del presente lavoro di tesi. In particolare, esse vengono classificate per tipo di richiesta e per dimensione dei dati scambiati (low, high) tra il client e il server.



gutta cavat lapidem

La goccia scava la pietra

Proverbio latino

Capitolo 4.

Analisi dei risultati sperimentali

Dopo un'accurata fase di progettazione e di caratterizzazione del workload per il caso di studio in esame, si è passati alla fase sperimentale. Sono stati eseguiti un certo numero di test preliminari, in particolare capacity test e test zero, utilizzando il test-bed a disposizione. Tali test sono stati eseguiti sia con il tool open source Jmeter che con un generatore ad-hoc. Dopo aver stabilito i valori dei livelli dei fattori nel DOE, sono stati predisposti i corrispondenti esperimenti e poi eseguiti con il generatore ad-hoc. In tali esperimenti sono state monitorate numerose variabili. A valle di un filtraggio dei valori di throughput del server MySQL e in congiunzione con la raccolta delle altre variabili monitorate è stata eseguita l'analisi. Da quest'ultima è possibile trarre importanti conclusioni sulla relazione tra il workload sottoposto al DBMS MySQL e il fenomeno del Software Aging.

4.1 Descrizione del Testbed

Il testbed rappresenta l'insieme delle attrezzature utilizzate per testare risorse hardware e/o applicazioni che vengono eseguite su di esse. In questo lavoro si è fatto utilizzo di un testbed fornito da SCOPE. Il Sistema Cooperativo distribuito ad alte Prestazioni per Elaborazioni Scientifiche ha come obiettivi:

- Software innovativo per la ricerca scientifica.
- Dati ad alte prestazioni e centro di elaborazione per applicazioni in contesti multidisciplinari.
- Infrastruttura GRID e middleware INFNGRID LGI/gLite.
- Compatibilità con il middleware EGEE.
- Integrazione nell'infrastruttura GRID europea e italiana.

Le aree di ricerca di SCOPE sono:

- Scienze del micro e del macro crosmo.
- Scienze dei materiali e dell’ambiente.
- Scienze biologiche.
- Scienze sociali.
- Middleware.

Per eseguire gli esperimenti oggetto dello studio sono state utilizzati due personal computers (client e server), opportunamente configurati in una rete LAN accessibile dall'esterno tramite protocollo ssh. La struttura della rete del testbed è illustrata dall'immagine 4.1:

4.2 Raccolta dei dati

Una volta scelte opportunamente la variabili da monitorare, sono stati raccolti i dati. Tra questi, per alcuni ne viene effettuato un opportuno filtraggio attraverso il calcolo delle statistiche delle variabili.

4.2.1 Monitoring

Un aspetto molto importante nell'esecuzione degli esperimenti è quella di individuare le varibili da monitorare. E' evidente che trascurare variabili che potrebbero dare importanti informazioni sullo stato e l'evoluzione del sistema può notevolmente compromettere l'analisi dei risultati.

Per ottenere il maggior numero di informazioni riguardanti lo stato del sistema, sono state selezionate inizialmente un gran numero di variabili da monitorare lato server. Successivamente con la tecnica Principal Component Analysis è stata analizzata l'informazione apportata da tali variabili. Le variabili monitorate sono quelle incluse in:

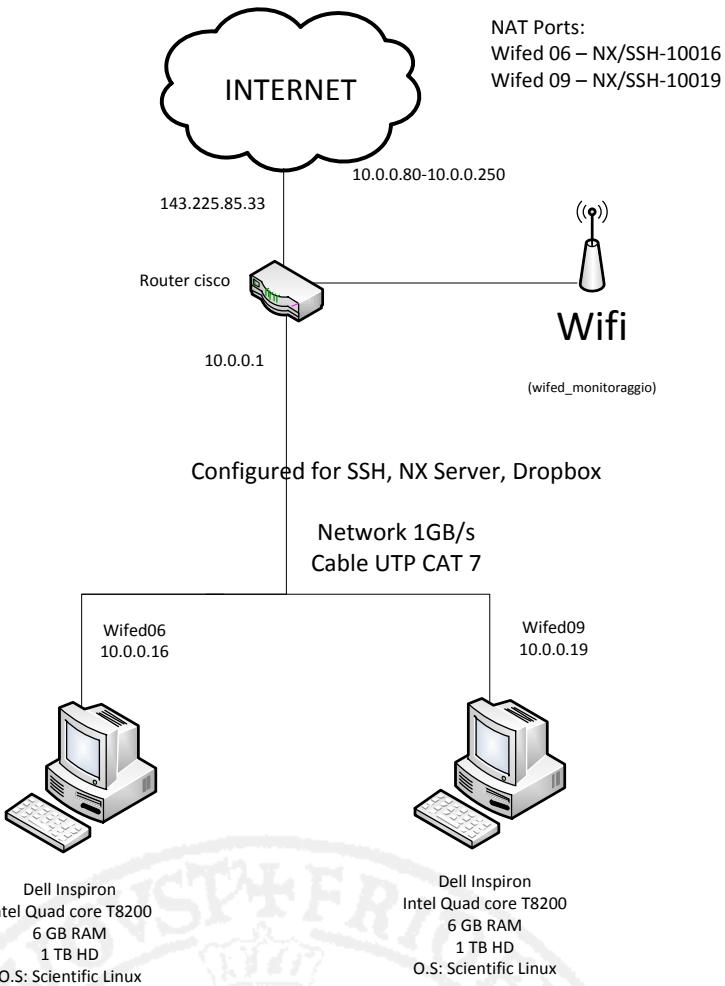


Figura 4.1: Test-bed utilizzato per gli esperimenti

- /proc/[pid del processo]/stat: riporta informazioni relative ai processi, la memoria, la paginazione della memoria, blocchi di I/O, interruzioni software e l'attività del processore.
- /proc/meminfo: riporta una grande quantità di informazioni importanti relative alla memoria del sistema operativo;
- sbin/sysctl fs.file-nr: lista degli handlers dei file nella memoria del kernel;
- ipcs: dà informazioni sui servizi Inter Process Communication per i processi chiamanti.

Lato client sono state monitorate le variabili:

- Timestamp: istante in cui viene servita correttamente la richiesta.
- Elapsed: il tempo che intercorre tra l'istante in cui viene fatta una richiesta e l'istante in cui essa viene servita correttamente.
- Latency: il tempo che intercorre tra la richiesta di connessione al database e l'istante in cui tale connessione avviene.

4.2.2 Filtraggio e preprocessing dei risultati

I risultati sperimentali ottenuti sia lato client che lato server sono stati opportunamente filtrati. In particolare sono stati eliminati gli outlier e per i dati lato client sono state calcolate le statistiche di media, deviazione standard, minimo, massimo mediante un'applicazione java. Nella fase precedente alla determinazione dei trend statistici, i dati sono stati anche normalizzati.

4.3 Esecuzione dei test preliminari

Prima di eseguire gli esperimenti è necessario elaborare alcuni test. I capacity test vengono eseguiti per determinare i limiti delle applicazioni in esame e i valori dei livelli da associare ai fattori. Il test zero è utile per stimare la durata dell' esperimento. Quest'ultima, infatti, dovrebbe essere pari ad un valore sufficiente ad osservare un significativo trend di aging nei dati sperimentali.

4.3.1 Capacity test

Dato un sistema, il capacity test sottopone il sistema ad un carico via via crescente, in termini di intensità e cioè numero di richieste nell'unità di tempo, fino a quando il sistema non riesce più a sostenere suddetto carico. In questo caso diverse situazioni sono possibili:

- Il sistema va in crash (blocco o improvvisa chiusura, non richiesta e inaspettata, di un programma). In tal caso il sistema non serve più le richieste.

- Nonostante venga aumentato gradualmente il numero di richieste al secondo sottoposte al sistema, quest'ultimo serve sempre lo stesso numero di richieste o un numero inferiore.

Durante i capacity test viene descritto l'andamento della curva che indica il numero di richieste soddisfatte dal sistema rispetto a quelle sottomesse. Se il sistema servisse correttamente tutte le richieste si osserverebbe una retta con pendenza 45° , nel caso di pendenza inferiore a 45° , invece, il sistema riesce a servire un numero di richieste inferiore a quello che dovrebbe. Il caso contrario non è ovviamente possibile. Nei sistemi reali tale retta non può avere pendenza 45° o inferiore in maniera illimitata, ad un certo punto nel grafico rappresentante tale fenomeno si osserverà un '*ginocchio*'. Il sistema, dopo aver servito correttamente un certo numero di richieste, va in uno stato di sovraccarico, non riuscendo più a gestire quello specifico livello di carico. Da quel punto in poi, il sistema serve sempre lo stesso numero di richieste o un numero decrescente di esse, nonostante il numero di richieste in ingresso, sottoposte al sistema, sia crescente.

4.3.1.1 Determinazione dei valori dei livelli

In fase sperimentale vengono associati i valori ai livelli presenti nel piano sperimentale realizzato.

Livelli intensità L'intensità delle richieste è caratterizzata da due livelli, low e high. Il caso low si riferisce al 30% del limite massimo di richieste nell'unità di tempo che il sistema riesce a soddisfare correttamente, mentre high il 70% di tale limite.

Livelli size La dimensione dei dati scambiati tra il client e il server è caratterizzata da due livelli, low e high. Poiché si vuole osservare se tale parametro influisca sulle dinamiche di aging, è importante che la differenza tra i due livelli sia significativa. A tal fine, si è scelto come dimensione low, l'occupazione di memoria dei dati pari a

1 Kilobyte mentre high 100 Kilobyte. I due livelli si differenziano di due ordini di grandezza.

Livelli tipologia di richiesta La tipologia di richieste è stabilita mediante l'uso di tecniche di clustering, secondo le quali le richieste vengono suddivise in due gruppi in base al tipo di carico che le stesse generano e cioè di tipo light o di tipo heavy. Come sopra descritto, per ottenere questo tipo di classificazione è stato utilizzando l'algoritmo di clustering k-means [1]. I dati raccolti dai capacity test, al variare dei tipi di richiesta e dei rate, sono stati sottoposti a tale algoritmo. Al fine di approfondire lo studio dei risultati ottenuti, per ciascun test sono state analizzate le seguenti variabili:

- Tasso di successo medio della richiesta: (tasso di risposta/tasso di richiesta).
- Deviazione standard del throughput (con throughput s'intende numero di richieste eseguite nell'unità di tempo).
- Mediana del throughput.
- Valore medio di elapsed (con elapsed s'intende il tempo che intercorre tra l'istante in cui viene fatta una richiesta e l'istante in cui essa viene servita correttamente).
- Deviazione standard di elapsed.
- Mediana di elapsed.
- Valore medio della latenza (con latenza s'intende il tempo che intercorre tra la richiesta di connessione al database e l'istante in cui tale connessione avviene).
- Deviazione standard della latenza.
- Mediana della latenza.

Nelle tecniche di analisi del cluster risultano di particolare importanza le operazioni iniziali di analisi dei dati. In particolare, i dati delle variabili raccolte sono state analizzate

Tabella 4.1: Cross-correlazione tra le variabili candidate alla clusterizzazione

Variabile 1	Variabile 2	Correlazione
ELAPSED-DEVSTD	THR-MEDIANA	0.7664
ELAPSED-DEVSTD	ELAPSED-M	0.9209
ELAPSED-MEDIANA	THR-MEDIANA	0.7782
ELAPSED-MEDIANA	ELAPSED-M	0.9787
ELAPSED-MEDIANA	ELAPSED-DEVSTD	0.912
LATENCY-DEVSTD	LATENCY-M	0.9052
LATENCY-MEDIANA	LATENCY-M	0.898
LATENCY-MEDIANA	LATENCY-DEVSTD	0.7346
ELAPSED-M	THR-MEDIANA	0.776
THR-MEDIANA	THR-DEVSTD	0.3932
ELAPSED-DEVSTD	SUCCESS-RATE	-0.3116
ELAPSED-M	THR-DEVSTD	0.304
LATENCY-DEVSTD	SUCCESS-RATE	-0.2927
LATENCY-DEVSTD	THR-DEVSTD	0.2921
ELAPSED-MEDIANA	THR-DEVSTD	0.2732
ELAPSED-DEVSTD	THR-DEVSTD	0.2723
ELAPSED-M	SUCCESS-RATE	-0.2628
LATENCY-M	THR-DEVSTD	0.2452
LATENCY-M	SUCCESS-RATE	-0.2272
LATENCY-MEDIANA	SUCCESS-RATE	-0.1892
ELAPSED-MEDIANA	SUCCESS-RATE	-0.1885
LATENCY-MEDIANA	THR-MEDIANA	-0.1785
LATENCY-MEDIANA	THR-DEVSTD	0.1516
LATENCY-M	THR-MEDIANA	-0.1393
LATENCY-MEDIANA	ELAPSED-MEDIANA	-0.1339
LATENCY-DEVSTD	ELAPSED-DEVSTD	0.1306
LATENCY-MEDIANA	ELAPSED-M	-0.1197
LATENCY-M	ELAPSED-MEDIANA	-0.1179
LATENCY-M	ELAPSED-M	-0.0869
LATENCY-MEDIANA	ELAPSED-DEVSTD	-0.0764
THR-DEVSTD	SUCCESS-RATE	-0.0714
THR-MEDIANA	SUCCESS-RATE	0.0429
LATENCY-DEVSTD	ELAPSED-M	0.0345
LATENCY-M	ELAPSED-DEVSTD	-0.0311
LATENCY-DEVSTD	ELAPSED-MEDIANA	-0.025
LATENCY-DEVSTD	THR-MEDIANA	-0.0008

dal punto di vista della cross correlazione, la correlazione a coppie, per avere un'idea di quali variabili risultano molto correlate tra di loro e cioè quali variabili fornirebbero informazioni duplicate o sovrapposte. Utilizzare due variabili che sono molto correlate e cioè con valore di cross correlazione $> |0, 7|$ in un'analisi di regressione o di un cluster potrebbe avere un impatto negativo sui risultati o comunque potrebbe alterare i risultati ottenuti dall'algoritmo di clustering. Il risultato dell'analisi di correlazione è riportato in tabella 4.1.

Come era facile intuire, i risultati della cross correlazione per le variabili in esame (success rate, elapsed, latency) mostrano la dipendenza tra le loro statistiche (media, deviazione standard, mediana).

In definitiva, all'algoritmo di clustering sono stati sottomessi solo i campioni relativi a:

- tasso di successo medio
- tempo elapsed medio

- tempo di latenza medio

Mediante i campioni ottenuti dai capacity test, eseguiti con tutti i tipi di richieste, con intensità crescente e dimensione dei dati high sono stati individuati i cluster indicati in tabella 4.2

Tabella 4.2: Clusters delle richieste

Richiesta	Cluster	
	Cluster 1	Cluster 2
Select	€	∅
Select nested	∅	€
Join	∅	€
Update-Insert-Delete	€	∅

Livelli variazione della richiesta Per la variazione di richiesta sono possibili due livelli: assenza o presenza di variazione a seconda se dopo una determinata richiesta venga eseguita la stessa o un' altra, scelta casualmente all'interno di un certo insieme, fissata la tipologia di richiesta.

Generazione del traffico per il DBMS Mysql Per eseguire i test preliminari e gli esperimenti pianificati deve essere utilizzato un generatore del traffico per DBMS. In particolare tale generatore produrrà un determinato carico sulla base dei valori dei fattori che si devono prendere di volta in volta in considerazione. Il carico per il DBMS è caratterizzato da un insieme di query, sottoposte al sistema con una certa intensità.

- Apache Jmeter

Si tratta di un'applicazione completamente scritta in Java [26], progettata per realizzare test di carico software client/server (ad esempio un'applicazione web).

Può essere usato per testare le prestazioni sia su risorse statiche che dinamiche, come file statici, servlet Java, script CGI, oggetti Java, database, server FTP, e altro ancora. JMeter può essere utilizzato per simulare un carico su un server o anche su una rete di calcolatori, per testarne le prestazioni complessive.

Per generare il carico, Apache Jmeter utilizza i *samplers*. I risultati generati da tali samplers possono avere diversi attributi (ad esempio successo/fallimento, tempo trascorso, dimensione dei dati, ecc..) che possono essere visualizzati attraverso i componenti *listeners*. Un particolare *sampler* è caratterizzato dalla richiesta JDBC (una query SQL) sottomessa ad un DBMS. Configurando opportunamente la connessione JDBC al database in esame è possibile indicare le query da eseguire, eventualmente secondo quale ordine o sotto quali condizioni. In una prima fase sono stati eseguiti i capacity test su MySQL utilizzando questo tool. Le richieste da sottomettere al DBMS sono state suddivise in quattro gruppi, come precedentemente descritto:

- query di visualizzazione (select semplici);
- query di visualizzazioni innestate (select nested);
- query di visualizzazione con congiunzione (join);
- query di modifica (update, insert, delete,...).

Fissata la dimensione dei dati scambiati tra il client e il server e il tipo di richiesta da testare, ogni test di capacità ha stressato il sistema MySQL con un carico via via crescente in termini di intensità delle richieste (numero di richieste nell' unità di tempo). Inizialmente si è considerato il caso di dimensione dei dati di tipo low e il carico sottomesso al sistema mediante Jmeter veniva suddiviso tra due client per simulare un caso più realistico.

I grafici in figura 4.2 mostrano nel caso di dimensione low l'andamento del reply rate, il numero di richieste servite nell'unità di tempo, rispetto al request rate, il numero di richieste sottomesse al sistema, nell'unità di tempo scelta:

Dai grafici si può notare che per tutti i tipi di richiesta, nel caso di dimensione low, il ginocchio si raggiunge con 150 richieste al secondo sottomesse.

Nel caso di dimensione dei dati di tipo high sono stati riscontrati alcuni problemi nell'effettuare i capacity test con il tool Jmeter. In particolare, già con un numero

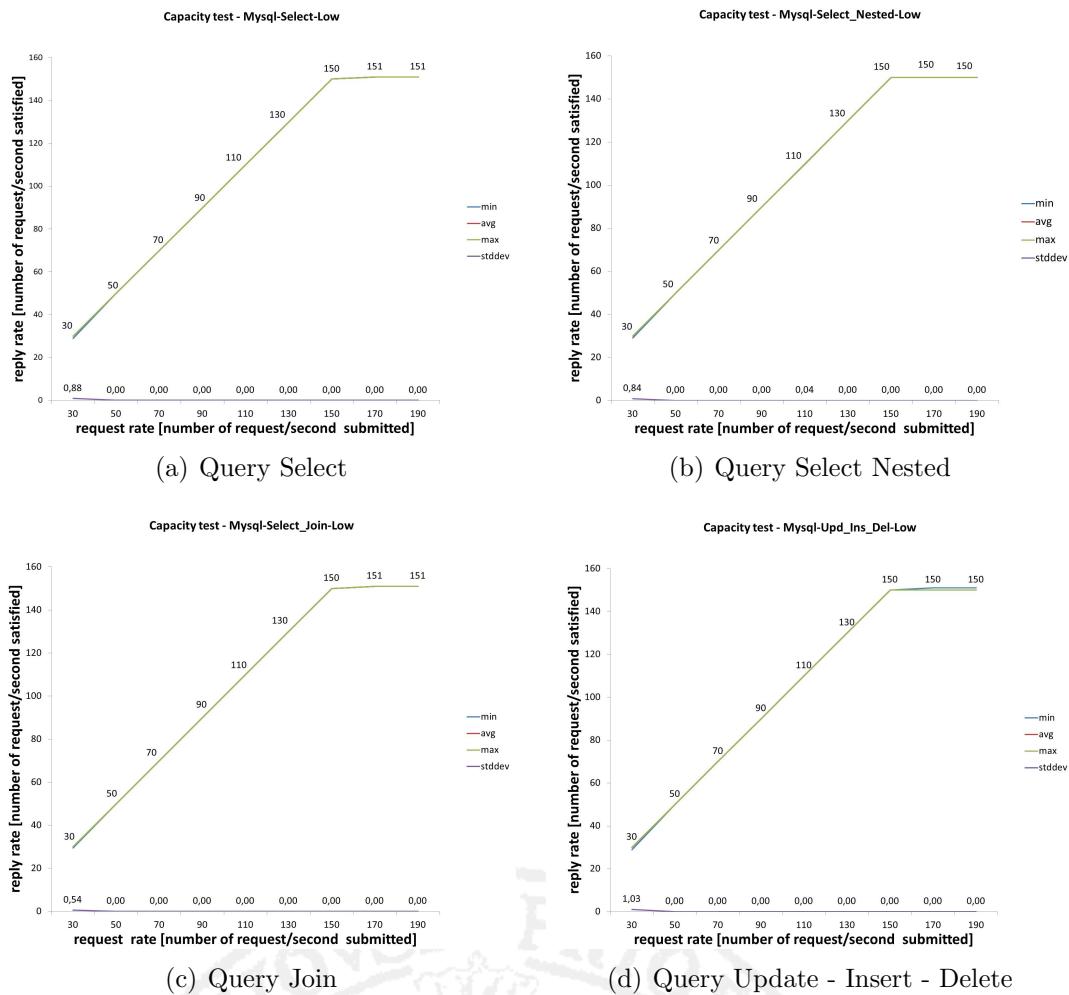


Figura 4.2: MySQL - Capacity test - Size low

molto basso di richieste sottomesse al server, il throughput del sistema risultava molto vicino allo zero. Approfondendo questo aspetto, si è notato che il sistema non riusciva a sostenere il throughput perché in media una richiesta durava più di un secondo e quindi quando arrivava una nuova richiesta con una già presente nel sistema, essa veniva accodata. Lo stesso avveniva anche per tutte le altre e quindi il sistema non riusciva a sostenere il rate richiesto. Sono state effettuate delle prove aumentando anche il numero di thread, fissato il rate, ma per sostenere quest'ultimo venivano esaurite le risorse del sistema.

Per risolvere tale problema la soluzione più opportuna è stata quella di realizzare un generatore ad-hoc.

L'applicazione java, quindi, ha lo scopo di sottomettere al sistema MySQL il numero di richieste desiderate. Viene generato un pool di thread ciascuno dei quali sottopone un certo tipo di richiesta al server. In questo modo i thread non vengono più accodati come con Jmeter.

I grafici in figura 4.3 mostrano nel caso di dimensione high l'andamento del reply rate, il numero di richieste servite nell'unità di tempo, rispetto al request rate, il numero di richieste sottomesse al sistema, nell'unità di tempo scelta

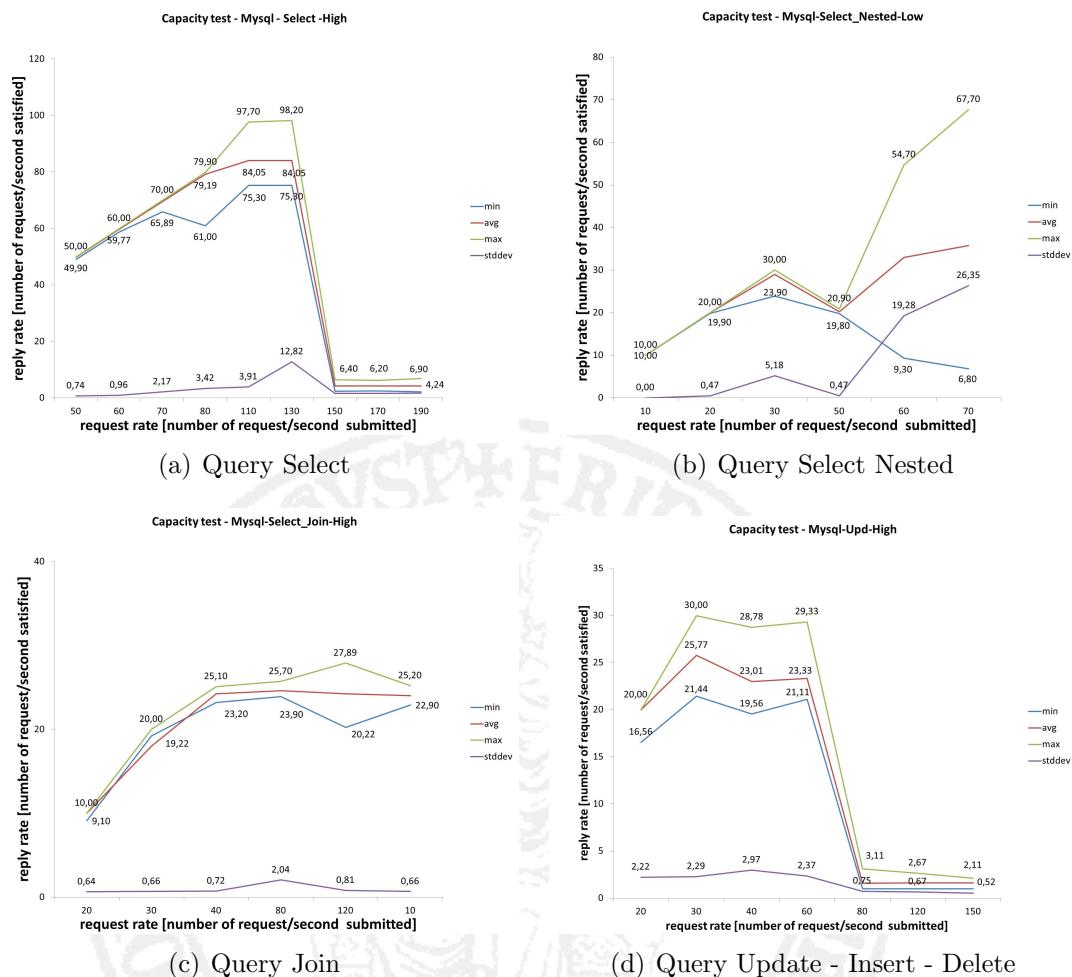


Figura 4.3: MySQL - Capacity test - Size high

4.3.2 Test zero

Il test zero ha lo scopo di individuare se è osservabile un trend di aging con il carico meno stressante, sottoposto al sistema. Questo test infatti viene eseguito settando i

parametri di workload al loro minimo livello.

In questo primo test sono state monitorate tutte le variabili descritte in /proc/meminfo relativamente alla distribuzione del sistema operativo prevista dal testbed in esame.

Si tratta di 30 variabili, in particolare Resident Set Size, la memoria utilizzata dal processo, la memoria libera, i buffers, la memoria cache utilizzata. Il test è durato 8 ore e costituisce una prima osservazione del comportamento del sistema MySQL.

Dai grafici 4.4 4.5 si può subito notare un aumento della memoria utilizzata dal processo e una lieve diminuzione della memoria libera del sistema.

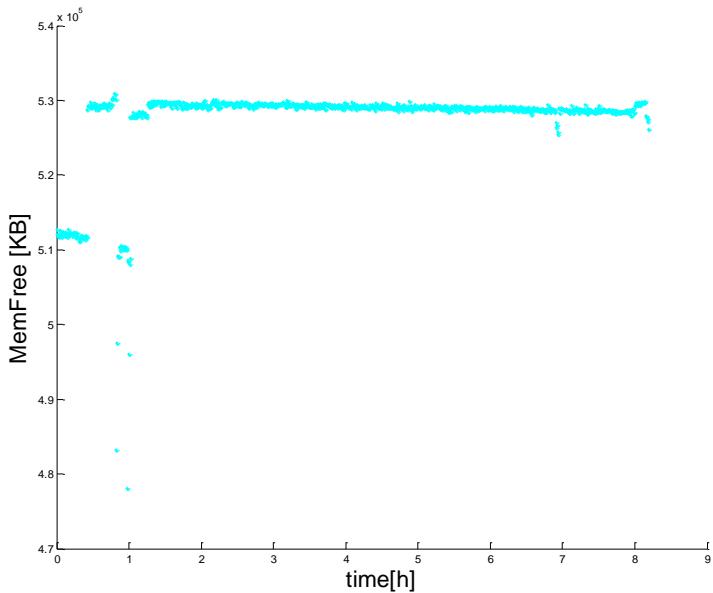


Figura 4.4: Test zero - Memoria libera del sistema

Poiché dai risultati ottenuti è evidente che 8 ore non bastano ad osservare, eventualmente, la presenza del fenomeno del Software Aging, si è eseguito il primo esperimento secondo la pianificazione DOE. La durata di tale esperimento è stata fissata a 24 ore.

Ai dati ottenuti è stato applicato l'algoritmo [22] per stimare il tempo minimo necessario ad osservare trend statisticamente significativi, fissata una certa variabile di risposta. Dato un livello di confidenza in input, esso stima se il numero di campioni collezionati fino ad un tempo t è sufficiente per ottenere un trend significativo. I risultati ottenuti hanno indicato come durata necessaria dell'esperimento 24 ore. Risulta fissata, dunque, la durata dei successivi esperimenti.

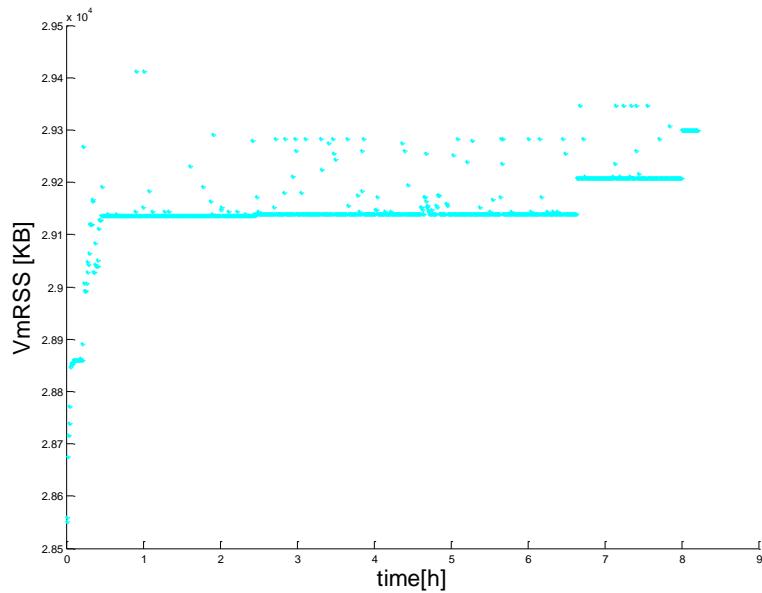


Figura 4.5: Test zero - Resident Set Size del processo

4.4 Esecuzione esperimenti

Utilizzando il generatore ad-hoc sono stati eseguiti 4 esperimenti, indicati nella tabella 4.3 sul sistema MySQL, ciascuno della durata di 24 ore.

Id esperimento	Nome Blocco	Intensità	Dimensione	Tipo di richiesta	Variazione
1	MySQL	Low	High	Light	YES
2	MySQL	High	High	Light	NO
3	MySQL	Low	Low	Heavy	NO
4	MySQL	High	High	Light	YES

Tabella 4.3: Piano sperimentale per MySQL

Nella tabella 4.4 sono invece indicati i valori specifici assegnati ai livelli.

Id esperimento	Nome Blocco	Intensità	Dimensione	Tipo di richiesta	Variazione
1	MySQL	15	100 Kylobyte	Light	YES
2	MySQL	56	100 Kylobyte	Light	NO
3	MySQL	45	1 Kylobyte	Heavy	NO
4	MySQL	36	100 Kylobyte	Light	YES

Tabella 4.4: Piano sperimentale per MySQL - valori specifici utilizzati

4.5 Analisi dei risultati

Una volta eseguiti gli esperimenti, vengono raccolti i risultati e dopo una fase di pre-processing e filtraggio si procede ad un'attenta fase di analisi per studiare i comportamenti del sistema in tutti i casi considerati e valutare la presenza di trend di Software Aging.

4.5.1 Preprocessing delle variabili monitorate

Nella fase di scelta delle variabili per il monitoring dello stato di un sistema, è di fondamentale importanza tener presente gli ordini di grandezza e le unità di misura implicate. Infatti, nel caso in cui gli ordini di grandezza e le unità di misura implicate siano molto diverse tra loro, al fine di effettuare in maniera corretta l'analisi, risulta necessario normalizzare i dati.

Un altro aspetto molto importante è rappresentato dalla correlazione tra le variabili.

Nella tabella 4.5 sono riportate le variabili con un valore di cross-correlazione > 0.90 .

Quelle, invece, con un valore di cross-correlazione > 0.70 e < 0.90 sono riportati in Tabella 4.6.

Nonostante si osservi un valore di cross-correlazione elevato è comunque importante approfondire il significato di ciascuna grandezza considerata, prima di escluderla in fase di analisi, in quanto eventualmente non necessaria. Si rimanda per tale approfondimento all'appendice C.

Dai risultati appena visti, però, è possibile osservare che non risulta opportuno considerare alcune variabili nell'analisi.

Si tratta delle variabili MemFree-LowFree, VmPeak-VmHWM. Infatti:

- LowFree=MemFree quando la quantità di memoria libera che è direttamente mappata nello spazio del kernel coincide con la memoria RAM fisica non utilizzata dal sistema. È il caso in cui tutta la memoria libera è direttamente mappata nello spazio del kernel.

Tabella 4.5: Cross-Correlazione Tra le variabili scelte per il monitoraggio con Corr > 0.90

Variabile1	Variabile2	Correlazione	Variabile 1	Variabile 2	Correlazione
VmSize	VmPeak	0,805301	Slab	MemFree	0,868927
VmHWM	VmPeak	0,789727	VmSize	Buffers	0,710985
VmRSS	VmPeak	0,770792	MemFree	Buffers	0,886318
VmPTE	VmPeak	0,801408	Cached	Buffers	0,86495
Committed_AS	VmPeak	0,709027	Active	Buffers	0,921695
VmPeak	VmSize	0,805301	LowFree	Buffers	0,886318
VmHWM	VmSize	0,84636	Slab	Buffers	0,778294
VmRSS	VmSize	0,790011	MemFree	Cached	0,939678
VmPTE	VmSize	0,753743	Buffers	Cached	0,86495
MemFree	VmSize	0,81711	Active	Cached	0,938428
Buffers	VmSize	0,710985	LowFree	Cached	0,939678
Active	VmSize	0,81255	Slab	Cached	0,88114
LowFree	VmSize	0,81711	VmSize	Active	0,81255
Committed_AS	VmSize	0,701464	MemFree	Active	0,969716
VmPeak	VmHWM	0,789727	Buffers	Active	0,921695
VmSize	VmHWM	0,84636	Cached	Active	0,938428
VmRSS	VmHWM	0,879803	LowFree	Active	0,969716
VmPTE	VmHWM	0,942281	Slab	Active	0,815203
VmPeak	VmRSS	0,770792	VmSize	LowFree	0,81711
VmSize	VmRSS	0,790011	MemFree	LowFree	1
VmHWM	VmRSS	0,879803	Buffers	LowFree	0,886318
VmPTE	VmRSS	0,882484	Cached	LowFree	0,939678
AnonPages	VmRSS	0,769054	Active	LowFree	0,969716
VmPeak	VmPTE	0,801408	Slab	LowFree	0,868927
VmSize	VmPTE	0,753743	VmRSS	AnonPages	0,769054
VmHWM	VmPTE	0,942281	MemFree	Slab	0,868927
VmRSS	VmPTE	0,882484	Buffers	Slab	0,778294
VmSize	MemFree	0,81711	Cached	Slab	0,88114
Buffers	MemFree	0,886318	Active	Slab	0,815203
Cached	MemFree	0,939678	LowFree	Slab	0,868927
Active	MemFree	0,969716	VmPeak	Committed_AS	0,709027
LowFree	MemFree	1	VmSize	Committed_AS	0,701464

Tabella 4.6: Cross-Correlazione Tra le variabili scelte per il monitoraggio con Corr > 0.70 e < 0.90

Variabile 1	variabile 2	Correlazione
VmPTE	VmHWM	0,942281
VmHWM	VmPTE	0,942281
Cached	MemFree	0,939678
Active	MemFree	0,969716
LowFree	MemFree	1
Active	Buffers	0,921695
MemFree	Cached	0,939678
Active	Cached	0,938428
LowFree	Cached	0,939678
MemFree	Active	0,969716
Buffers	Active	0,921695
Cached	Active	0,938428
LowFree	Active	0,969716
MemFree	LowFree	1
Cached	LowFree	0,939678
Active	LowFree	0,969716

- VmPeak-VmHWM rappresentano rispettivamente il picco della dimensione della memoria virtuale e picco della memoria utilizzata dal processo (VmRSS). Essendo entrambi legati a VmRSS risulta più opportuno non includerli nell'analisi dei dati.

Per tutte le altre variabili tali affermazioni non risultano così scontate, per cui risulta più giusto includerle comunque nell'analisi.

4.5.2 Principal Component Analysis

Dato il numero elevato di variabili monitorate è stata eseguita un' analisi dei componenti principali o anche detta PCA. Si tratta di un metodo non parametrico per estrarre informazioni rilevanti da un insieme di dati contenente rumore. Il suo obiettivo principale è ridurre la dimensionalità di un dataset contenente un grande numero di variabili correlate, eliminando la ridondanza dell'informazione e allo stesso tempo conservare il più possibile la variazione presente nel dataset iniziale [1].

Geometricamente, l'obiettivo della PCA è presentare i dati nel riferimento che evidenzia maggiormente la loro struttura (cambio di riferimento), come rappresentato in figura 4.7. Il cambio di riferimento può essere visto come un cambio di punto di vista che massimizza l'informazione visibile nei dati, come riportato in figura 4.6[1].

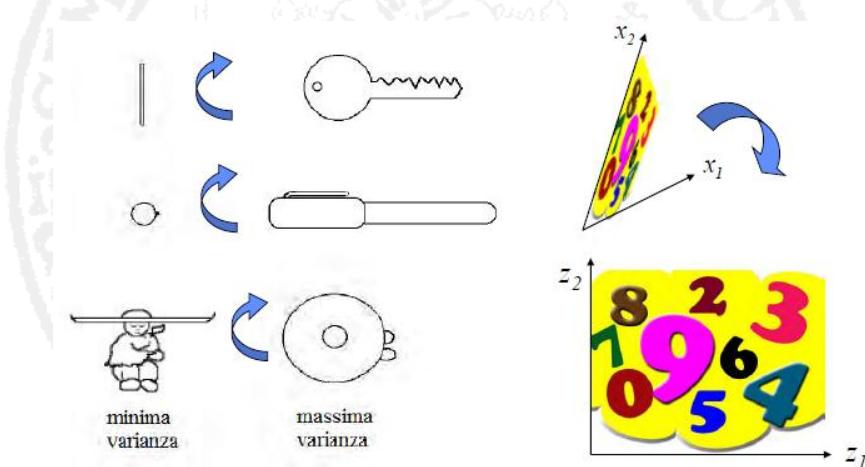


Figura 4.6: Esempio di variazione di punti di vista per massimizzare l'informazione visibile su un fenomeno. Immagine tratta da [1]

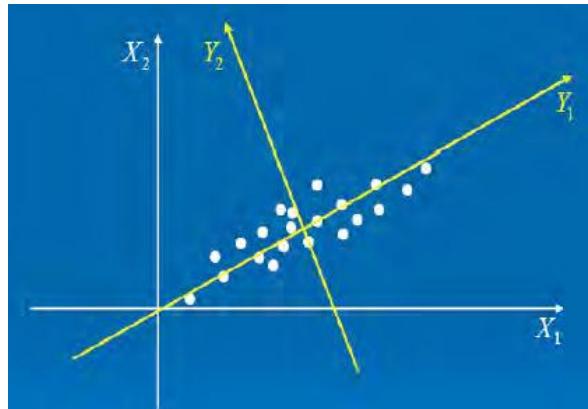


Figura 4.7: Esempio di cambio di riferimento mediante PCA

I componenti principali (PCs), rappresentanti gli assi del nuovo sistema riferimento, sono nuove variabili ottenute come combinazione lineare delle variabili originali, ottenute in modo che siano ortogonali tra loro, cioè incorrelate. Inoltre, esse sono ordinate in modo tale che le prime rappresentano la maggior parte della variabilità dei dati presente in tutto il dataset originale.

Dato un dataset X , rappresentato in figura 4.8, i PCs sono definiti come di seguito. Sia $v = (v_1, v_2, \dots, v_p)'$ un vettore di p variabili casuali dove $'$ è l'operazione di trasposizione.

$$\mathbf{x} = \left[\begin{array}{cccc} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{array} \right] \quad \left. \right\} n \text{ misure } \mathbf{x} \in \mathbb{R}^{n \times p}$$

Figura 4.8: Esempio di dataset

Il primo passo è trovare una funzione lineare $a'_1 v$ degli elementi di v che massimizzano la varianza dove a_1 è un vettore p -dimensionale $(a_{11}, a_{12}, \dots, a_{1p})'$, così da:

$$a'_1 v = \sum_{i=1}^p a_{1i} v_i \quad (4.1)$$

Dopo aver identificato $a'_1 v, a'_2 v, \dots, a'_{j-1} v$ viene cercata una funzione lineare $a'_j v$ che è

incorrelata con $a'_{j-1}v$ e che abbia la massima varianza. Di seguito vengono trovate d funzioni lineari dopo d passi. La j -esima variabile $a'_j v$ è la j -esima PC. In generale, la maggior parte della variazione di v verrà tenuta in conto maggiormente da poche prime componenti dei PCs.

Per trovare la forma di tutte le PCs, dobbiamo conoscere la matrice di covarianza Σ di v . La matrice di covarianza rappresenta il modo in cui ogni variabile varia rispetto alle altre. Nel caso considerato del dataset X , formato da n vettori di lunghezza p , la matrice di covarianza avrà perciò dimensioni $p \times p$ e i suoi valori sono definiti come:

$$\sigma_{ij} = \frac{1}{n} \sum_{h=1}^n (x_{hi} - \mu_j)^2 \quad (4.2)$$

Dove ogni x_{hi} rappresenta una caratteristica (il vettore x rappresenta una rilevazione), il vettore μ la media dei valori di ciascuna caratteristica (perciò μ_j rappresenta la j -esima caratteristica), e n è il numero di rilevazioni.

I valori sulla diagonale σ_{ij} con $i = j$, rappresentano la varianza dei dati in input. Ogni elemento σ_{ij} con $i \neq j$, rappresenta la correlazione tra i e j. Nel caso in cui questo valore sia positivo, significa che al crescere di una caratteristica, statisticamente cresce anche l'altra. Nel caso in cui questo valore sia negativo, accade il contrario. Se le caratteristiche sono statisticamente indipendenti, questo valore è 0 (l'implicazione inversa non è necessariamente verificata).

Generalmente quando i vettori del dataset X hanno ordini di grandezza molto diversi, si preferisce svolgere la PCA su dati standardizzati utilizzando lo Z-score, cioè dati a media nulla e varianza unitaria, ottenuti mediante la seguente trasformazione:

$$z = \frac{x - \bar{x}}{\sigma} \quad (4.3)$$

dove σ è la varianza dei campioni iniziali. In questo caso la matrice di covarianza

si riduce per costruzione alla matrice di correlazione, ovvero una matrice dove ogni elemento c_{ij} rappresenta la correlazione del vettore i verso il vettore j del dataset X , e che risulta dunque simmetrica rispetto alla diagonale formata da tutti elementi unitari¹.

In sintesi il primo passo $z_1 = a'_1 v$, dove z_1 è il primo PC, può essere identificato risolvendo il seguente problema di ottimizzazione:

$$\max var(a'_1 v) \text{ soggetto a } a'_1 a = 1 \quad (4.4)$$

dove $var(a'_1 v)$ è calcolata come

$$vara'_1 v = a'_1 \Sigma a_1 \quad (4.5)$$

La trasformazione lineare ortogonale del dataset mediante la base costituita dai PCs consente di ottenere due risultati principali. Il primo riguarda la **Feature Selection**, ovvero classificare le caratteristiche importanti del dataset X secondo la loro importanza. Il secondo riguarda la riduzione delle dimensioni, ovvero quantificare la perdita di informazione derivante dall'eventuale riduzione della dimensionalità dei dati del dataset X . In particolare, la PCA consente di quantificare la percentuale di informazione nelle varie componenti ordinate per importanza, in modo da conoscere la perdita di informazione per ciascuna componente esclusa dalla riduzione. Ulteriori dettagli su come tale sistema viene risolto utilizzando la tecnica degli autovettori sono riportati in [1].

La PCA fornisce una spiegazione alternativa alla variabilità osservata con il pregio di descrivere il fenomeno oggetto di studio mediante dimensioni tra loro non correlate e ordinate in termini della loro importanza nella spiegazione. Questo permette di interpretare il fenomeno attraverso il nuovo significato assunto dalle componenti

¹la correlazione di un elemento verso se stesso è 1 per definizione.

principali che non sono state scartate dalla PCA, e ridurre il numero di variabili considerate, scartando quelle che contribuiscono poco alla variabilità osservata (cioè quelle con covarianza bassa).

Una volta eliminati gli outlier per tutti gli esperimenti, i valori delle variabili monitorate vengono normalizzati e poi sottoposti all'algoritmo che esegue la PCA, ad eccezione delle variabili che presentavano, come detto prima, un'alta correlazione con le altre. La figura 4.9 mostra la distribuzione dei valori delle variabili.

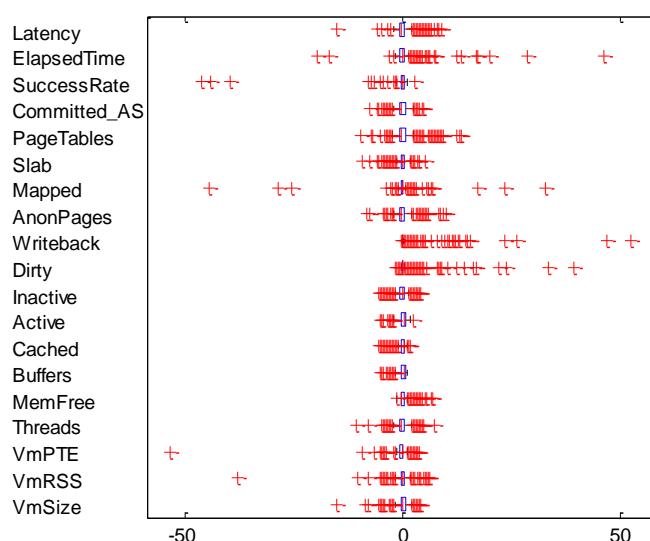


Figura 4.9: Dispersione delle variabili monitorate

La Principal component Analysis fornisce:

- W, la matrice degli autovettori.
- Z, le osservazioni Z-trasformate dei dati nel riferimento PCA.
- L, gli autovalori, ordinati in maniera crescente.

Ciò che in prima analisi si vuole osservare è come le variabili si ‘muovano complessivamente’, se esistono, cioè, delle variabili secondo cui il fenomeno oggetto di studio possa essere rappresentato mediante dimensioni tra loro non correlate. Una volta individuate

le matrice degli autovettori, dei dati nel nuovo riferimento e il vettore degli autovettori, risulta molto interessante valutare la varianza spiegata da ciascuna componente indicata nella tabella 4.7.

Tabella 4.7: Valori della varianza spiegata

VAR SPIEGATA [%]
42,4058
9,5626
8,2426
7,3320
5,6355
5,1727
4,5600
4,3058
3,8766
3,2130
1,8454
1,1528
0,9442
0,5573
0,4919
0,3308
0,1632
0,1532
0,0545

Da questa tabella e dalla figura 4.10 è possibile notare che non esistono una o due componenti che spiegano completamente il dataset. Questa osservazione porta a non poter considerare soltanto una o due componenti nell' analisi PCA. Ad ogni modo, per non rischiare di allargare troppo il campo di osservazione, è possibile considerare le prime quattro componenti, le più importanti.

Nell'analisi PCA vengono calcolati, per ciascuna variabile, i valori degli autovettori. Le prime cinque componenti vengono analizzate nella tabella 4.8 al fine di valutare quali variabili contribuiscono e con quale peso all'informazione contenuta nell' intero dataset.

Nell'analisi PCA è importante valutare il coefficiente per ciascuna variabile analizzata, infatti le variabili con coefficiente più alto sono quelle più rilevanti per il fenomeno, come spiegato in precedenza.

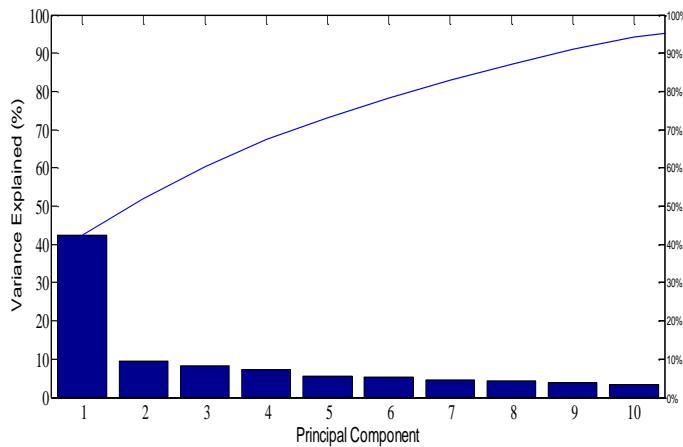


Figura 4.10: Grafico di Pareto della varianza spiegata

Tabella 4.8: Coefficienti analisi PCA delle prime cinque componenti

	PC1	PC2	PC3	PC4	PC5
'VmSize'	2,9467E-01	2,4254E-01	-3,7822E-02	-1,5095E-01	-2,1686E-03
'VmRSS'	2,5851E-01	3,3231E-01	1,2506E-02	-2,5782E-01	8,1772E-02
'VmPTE'	2,5737E-01	3,5210E-01	3,9896E-02	-1,9931E-01	5,8438E-02
'Threads'	1,0523E-01	3,4540E-01	2,8351E-02	8,6491E-02	1,4099E-01
'MemFree'	-3,2150E-01	1,8193E-01	1,8522E-01	-1,0551E-01	-6,5646E-02
'Buffers'	3,0531E-01	-6,7266E-02	-2,3555E-01	4,8325E-02	-6,4694E-02
'Cached'	2,9564E-01	-2,5262E-01	-2,5770E-01	1,1503E-01	3,2815E-02
'Active'	3,3324E-01	-1,1654E-01	-1,4423E-01	8,1950E-02	-4,2605E-02
'Inactive'	-8,1616E-02	-2,6067E-02	-1,1673E-01	3,2327E-01	7,4332E-01
'Dirty'	-6,1886E-03	-1,5288E-01	4,6890E-01	4,1102E-02	4,5252E-02
'Writeback'	5,2556E-03	-6,4935E-02	2,9557E-01	1,0924E-01	2,2739E-01
'AnonPages'	3,1366E-01	3,7657E-02	2,4893E-01	-4,0469E-02	-2,2539E-02
'Mapped'	1,3484E-01	-2,2209E-01	4,6646E-01	1,7994E-01	1,1380E-01
'Slab'	3,0208E-01	-2,3083E-01	-1,7273E-01	1,6560E-01	1,0764E-01
'PageTables'	2,3919E-01	-1,7284E-01	3,6920E-01	4,4556E-02	-1,7392E-01
'Committed_AS'	3,1608E-01	6,9385E-02	1,8781E-01	1,9421E-02	-2,0080E-02
'SuccessRate'	-4,3903E-02	-3,8703E-02	-1,0154E-01	4,9259E-01	-4,6003E-01
'ElapsedTime'	1,0411E-02	3,5451E-01	7,8914E-02	4,3551E-01	-2,6250E-01
'Latency'	1,7434E-02	4,2225E-01	2,5550E-02	4,6462E-01	1,2695E-01

Come è facile osservare, tutte le variabili danno un contributo ad almeno una delle componenti principali considerate, non c'è alcuna variabile che non partecipa al fenomeno, senza un peso rilevante. Grazie a questo importante risultato è possibile affermare che risulta *necessario* analizzare il comportamento di tutte queste variabili. Per approfondire l'analisi si è fatto riferimento alle prime due componenti, che presentano la varianza spiegata maggiore. In 4.11 sono riportati i valori delle variabili monitorate per ciascun esperimento, nel nuovo riferimento e i coefficienti rispetto alle due componenti principali.

Ciascuna delle 16 variabili è rappresentata in questo grafico attraverso un vettore, la cui direzione e lunghezza indica quanto ciascuna variabile contribuisce alle due componenti

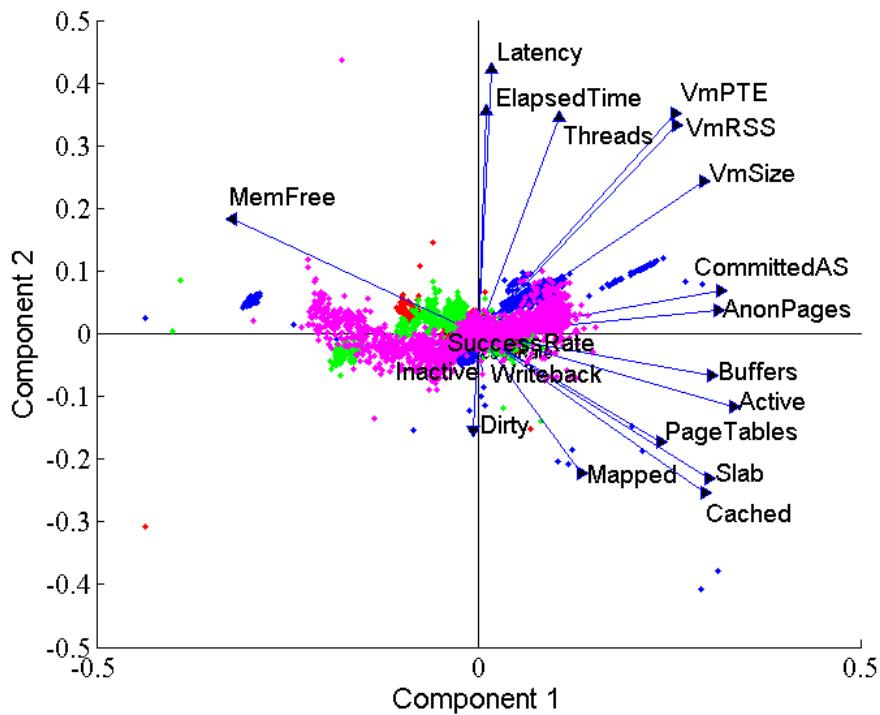


Figura 4.11: Analisi PCA - coefficienti delle variabili monitorate e osservazioni Z-trasformate dei dati nel nuovo riferimento

principali. Per esempio è possibile vedere che la prima componente, rappresentata in questo grafico dall'asse orizzontale, presenta i coefficienti positivi per tutte le variabili ad eccezione della variabile MemFree (memoria libera del sistema). Come indicato in 4.12 questo corrisponde ai 15 vettori diretti nella metà destra del grafico e l'unico vettore nella metà sinistra. Quindi la prima componente permette di dividere il dataset in due parti:

- Dati che presentano memoria libera alta \Rightarrow assenza aging del software o comunque probabilità molto bassa che il fenomeno sia presente.
- Dati che presentano memoria libera bassa e le altre variabili basse \Rightarrow potenziale presenza di aging del software.

La seconda componente, invece, rappresentata dall'asse verticale, ha i coefficienti positivi per le variabili: MemFree(coefficiente basso), Latency, ElapsedTime, Threads, VmPTE, VmRSS, VmSize, CommittedAS, AnonPages e negativi per le restanti varia-

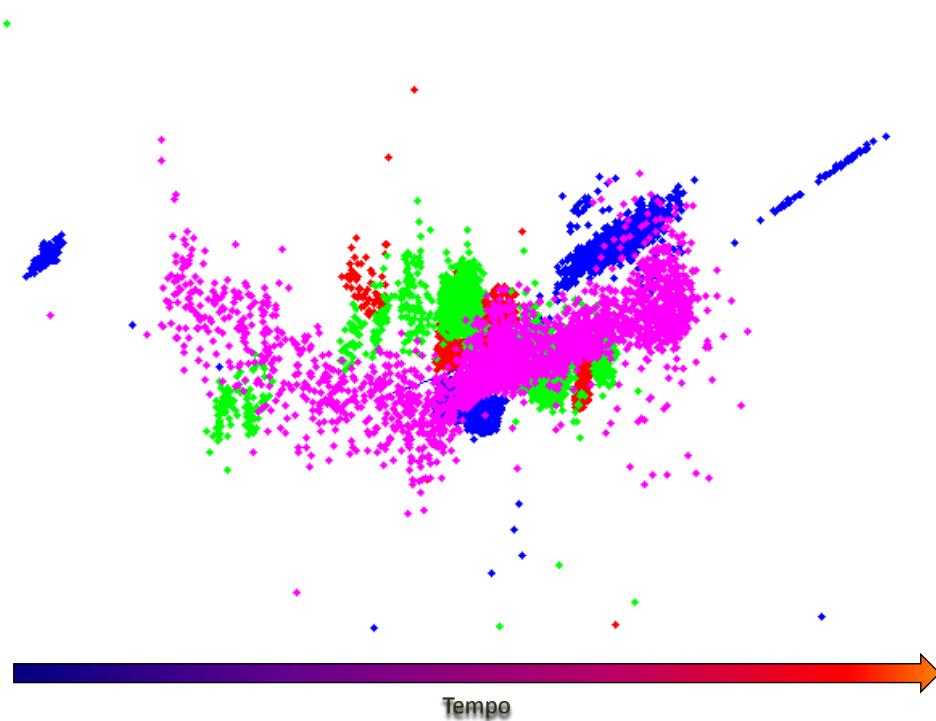


Figura 4.12: Relazione dati nel nuovo riferimento - ore trascorse per la prima componente.

bili e cioè Buffers, Active, PageTables, Slab, Cached, Mapped. Come mostrato in 4.13 questo corrisponde ai vettori diretti rispettivamente nella metà superiore e inferiore del grafico ed indica che la componente in questione, la seconda, distingue tra i dati sperimentali che hanno valori alti per il primo sottoinsieme di variabili e valori bassi per il secondo sottoinsieme e viceversa i dati sperimentali che hanno valori bassi per il primo sottoinsieme di variabili e valori alti per il secondo sottoinsieme.

Le restanti variabili non danno un grosso contributo informativo e cioè SuccessRate, Inactive, WriteBack, Dirty.

4.5.2.1 Risultati dell' analisi

Concludendo l'analisi dei componenti principali (PCA), si possono riassumere le osservazioni più salienti:

- L'evidente schiacciamento dei dati lungo asse delle ascisse (componente 1) mostra una forte dipendenza tra i dati monitorati e la variabile MemFree (che presenta

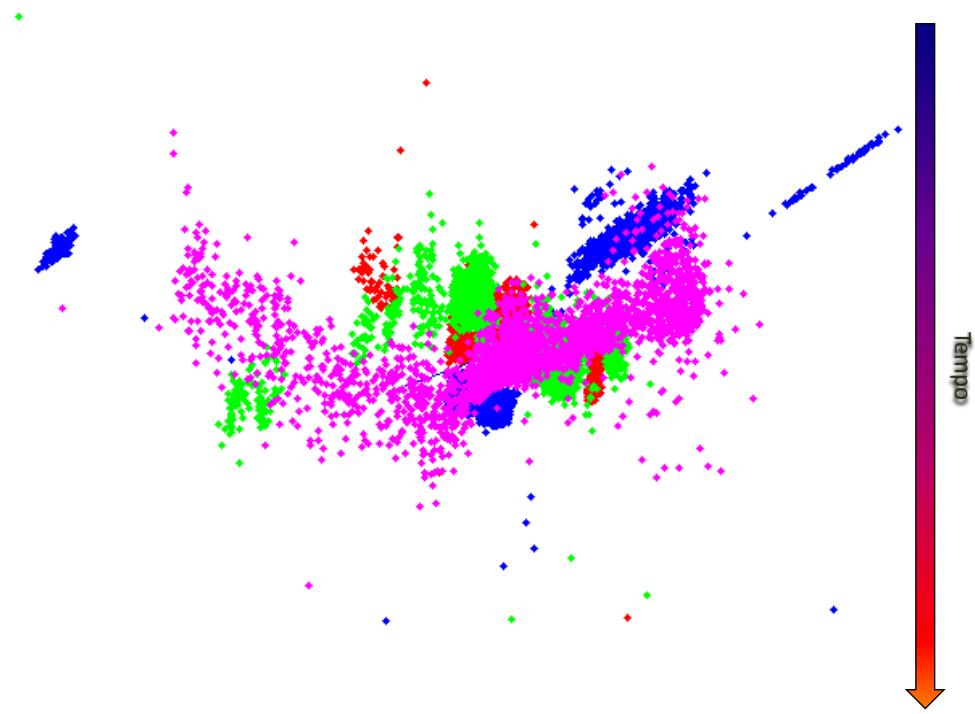


Figura 4.13: Relazione dati nel nuovo riferimento - ore trascorse per la seconda componente.

un contributo molto rilevante rispetto alla componente 1)

- I dati ottenuti da ciascun esperimento che sono presenti nel secondo e terzo quadrante sono relativi alle prime ore degli esperimenti, mentre quelli nel primo e quarto quadrante sono relativi alle ultime ore degli esperimenti 4.12. Se si considera la variabile MemFree, che dà un grosso contributo sulla prima componente, rappresentata dall'asse delle ascisse, è evidente che i dati relativi alle prime ore presentano valori di memoria libera maggiori rispetto a quelli nelle ultime ore. Questo dunque è un indicatore di aging del sistema. Analogamente per le variabili CommittedAS, AnonPages, Buffers, PageTables, Slab, Cached che però, avendo coefficiente positivo per la prima componente, saranno minori nelle prime ore sperimentali e maggiori nelle ultime ore sperimentali. Si tratta di un comportamento coerente, se infatti la memoria libera diminuisce è ovvio che sarà allocato maggiore spazio per i buffer o per la memoria cache utilizzata dal kernel.

- Allo stesso modo, se si osserva la seconda componente 4.11, i dati si distribuiscono lungo l'asse delle ordinate, nel verso delle ore sperimentali crescenti dal basso verso l'alto. Questo accade solo per gli esperimenti 2 e 4. Per questi ultimi, dunque, all'aumentare delle ore sperimentali si osserva che aumentano le variabili Elapsed, Latency, Threads, VmRSS, VmSize e quindi il tempo di servizio (elapsed) cresce al crescere di tali variabili. Per gli esperimenti 1 e 3 si vedrà successivamente che non sono stati rilevati trend per tali variabili, questo spiega il comportamento appena descritto.
- Da tutte queste osservazioni è possibile ricavare importanti conclusioni sull'influenza di ciascuna variabile di Workload sul fenomeno di aging. Per tutti gli esperimenti è stato osservata l'influenza sul fenomeno delle variabili MemFree, CommittedAS, AnonPages, Buffers, PageTables, Slab, Cached. Per il secondo ed il quarto esperimento è stata osservata l'influenza delle variabili Elapsed, Latency, Threads, VmRSS, VmSize, Mapped, per il primo ed il terzo questo non accade. Riprendendo il piano sperimentale, è possibile notare che il primo e il terzo esperimento differiscono dal secondo e il quarto per l'intensità delle richieste. Dunque un *fattore che sicuramente influisce sul fenomeno del Software Aging è l'intensità*.
- Se si sconsiglia l'esperimento che prevedeva maggiore intensità, il secondo (56 richieste/s), è possibile osservare dal grafico precedente che tale esperimento mostra la massima variazione (alta-bassa) della variabile MemFree quindi *influisce molto sulle dinamiche del fenomeno*.

Concludendo, osservando i dati sperimentali e analizzando approfonditamente le variabili monitorate è possibile osservare che:

- Non basta monitorare soltanto le variabili relative a memoria libera e prestazioni del sistema perché dai risultati ottenuti è evidente che altre le altre variabili giocano un ruolo importante nella comprensione del fenomeno del Software Aging.

- Non sempre le variabili monitorate sono ben modellate con tecniche di regressione lineare.

4.5.3 Determinazione dei trend di Aging

Come in 2.4 un test molto adottato per determinare terminare la presenza di aging è quello di Mann Kendall e il metodo di Sen per la stima della pendenza. L'ipotesi nulla da testare è: $H_0 = \text{assenza di aging}$. Attraverso uno script matlab viene realizzato tale test. Tra tutte le variabili in output la seguente tabella mostra quelle più significative, e cioè p-value, sen, cilower e ciupper per tutti gli esperimenti effettuati. In particolare, il p-value è la probabilità di osservare un dato risultato nelcaso in cui il test di ipotesi nulla sia vero. Si rifiuterà l'ipotesi nulla (assenza di aging) quando il p-value risulta minore del livello di significatività scelto. Quest'ultimo è stato fissato a 0.05. Sen è la pendenza della retta che rappresenta il trend, cilower e ciupper l'intervallo di confidenza per sen. I risultati ottenuti per le variabili indicative dello stato della memoria per ciascun esperimento sono mostrati nelle tabelle 4.9 4.104.114.12:

Da tali tabelle è possibile osservare le variabili che presentano (segnate con il rosso) o non presentano (segnate con l'azzurro) un trend. In particolare:

Esperimento 1: Assenza di trend per le variabili: VmPeak, VmSize, VmPTE, Threads, Writeback. Trend molto significativo ($|sen| > 4$ e $|sen| < 8$): MemFree,Cached,Active,LowFree.

Esperimento 2: Assenza di trend per le variabili: Dirty, Writeback, NumOfAllocFH.

Trend abbastanza significativo ($|sen| > -1.41$ e $|sen| < 2$): MemFree,Buffers,Active,LowFree.

Esperimento 3: Assenza di trend per le variabili: VmPeak, VmHWM, VmPTE, Threads,Dirty, Writeback, Mapped, NumOfAllocFH. Trend poco significativo ($|sen| > -0.13$ e $|sen| < 0.27$): MemFree,Inactive,Active,LowFree,Committed_AS.

Esperimento 4: Assenza di trend per le variabili: Threads, Writeback, NumOfAllocFH.

Trend abbastanza significativo ($|sen| > -2.07$ e $|sen| < 2.50$): MemFree,Cached,Active,LowFree.

I risultati ottenuti per le variabili indicative delle prestazioni del sistema per ciascun esperimento sono mostrati nella tabella 4.13

Tabella 4.9: Risultati Mann-Kendall test - Esperimento 1 - Variabili relative allo stato della memoria

Nome variabile	Id_Exp	1			
		p-value	sen	Clower	Cupper
ASSENZA					
'VmPeak'	TREND	0,0000E+00	0,0000E+00	0,0000E+00	
'VmSize'	2,6837E-148	0,0000E+00	0,0000E+00	9,3305E-05	
'VmHWM'	0,0000E+00	5,7770E-04	0,0000E+00	6,2539E-04	
'VmRSS'	1,8865E-08	-1,0225E-04	-1,8157E-04	0,0000E+00	
'VmPTE'	8,3470E-02	0,0000E+00	0,0000E+00	0,0000E+00	
'Threads'	2,0074E-02	0,0000E+00	0,0000E+00	0,0000E+00	
'MemFree'	0,0000E+00	-7,2785E+00	-7,4457E+00	-7,0970E+00	
'Buffers'	0,0000E+00	3,6088E-01	3,5004E-01	3,7255E-01	
'Cached'	0,0000E+00	4,3147E+00	4,2648E+00	4,3652E+00	
'Active'	0,0000E+00	7,8598E+00	7,7717E+00	7,9475E+00	
'Inactive'	7,3558E-284	-8,7170E-01	-8,9901E-01	-8,4626E-01	
'LowFree'	0,0000E+00	-7,2785E+00	-7,4457E+00	-7,0970E+00	
'Dirty'	1,1218E-24	4,2283E-04	3,3085E-04	5,2317E-04	
'Writeback'	2,0750E-02	0,0000E+00	0,0000E+00	0,0000E+00	
'AnonPages'	1,5959E-270	1,4341E-02	1,2622E-02	1,6931E-02	
'Mapped'	0,0000E+00	2,2355E-03	2,0880E-03	2,4180E-03	
'Slab'	0,0000E+00	8,0000E-02	7,4608E-02	8,6545E-02	
'PageTables'	5,7879E-221	2,9963E-04	2,2870E-04	4,0963E-04	
'Committed_AS'	'	0,0000E+00	1,5514E-01	1,3819E-01	1,7516E-01
'NumOfAllocFH'	'	2,1674E-237	1,8124E-02	1,5726E-02	2,0055E-02

Da tali tabelle è possibile osservare le variabili che presentano un lieve trend o non presentano trend (segnate con l'azzurro). In particolare:

Esperimento 1: Assenza di trend per le variabili: Success Rate, Elapsed Time. Trend poco significativo ($sen = 9.1369E - 07$): Latency Time.

Esperimento 2: Assenza di trend per le variabili: Success Rate. Trend poco significativo ($sen = 6.3275E - 06$ e $sen = 2.0533E - 07$): Elapsed Time,Latency Time.

Esperimento 3: Assenza di trend per le variabili: Success Rate, Elapsed Time,Latency Time.

Esperimento 4: Assenza di trend per le variabili: Success Rate. Trend poco significativo ($sen = 4.5352E - 06$ e $sen = 7.3330E - 07$): Elapsed Time,Latency Time.

Da questa prima analisi è evidente che il workload del terzo esperimento risulta molto meno stressante per il sistema. L'esperimento è stato settando con i parametri:

Intensità: 30% della capacità massima del sistema.

Dimensione: 1 Kylobyte (Low).

Tabella 4.10: Risultati Mann-Kendall test - Esperimento 2 - Variabili relative allo stato della memoria

Nome variabile	Id_Exp			
	p-value	sen	Clower	Cupper
'VmPeak'	0,0000E+00	3,7760E-01	3,6603E-01	3,8973E-01
'VmSize'	0,0000E+00	2,9270E-01	2,8322E-01	3,0279E-01
'VmHWM'	0,0000E+00	1,8560E-01	1,8097E-01	1,9076E-01
'VmRSS'	0,0000E+00	1,2456E-01	1,2020E-01	1,2886E-01
'VmPTE'	0,0000E+00	9,9875E-04	9,6899E-04	1,0303E-03
'Threads'	2,6173E-222	9,6398E-04	9,1324E-04	1,0148E-03
'MemFree'	0,0000E+00	-1,4103E+00	-1,4242E+00	-1,3968E+00
'Buffers'	0,0000E+00	1,1111E+00	1,0918E+00	1,1292E+00
'Cached'	0,0000E+00	1,7207E-02	1,7171E-02	1,7248E-02
'Active'	0,0000E+00	1,4439E+00	1,4387E+00	1,4493E+00
'Inactive'	8,1350E-72	-2,2908E-01	-2,4256E-01	-2,0951E-01
'LowFree'	0,0000E+00	-1,4103E+00	-1,4242E+00	-1,3968E+00
'Dirty'	2,0059E-02	0,0000E+00	-4,8362E-05	0,0000E+00
'Writeback'	4,9872E-02	0,0000E+00	0,0000E+00	0,0000E+00
'AnonPages'	0,0000E+00	1,2392E-01	1,1964E-01	1,2836E-01
'Mapped'	0,0000E+00	2,4876E-04	2,1231E-04	3,1373E-04
'Slab'	0,0000E+00	5,7485E-02	5,5587E-02	5,9686E-02
'PageTables'	0,0000E+00	9,1614E-04	8,8300E-04	9,4821E-04
'Committed_AS'	'	0,0000E+00	3,2979E-01	3,1904E-01
'NumOfAllocFH'		8,9831E-133	0,0000E+00	0,0000E+00

Tabella 4.11: Risultati Mann-Kendall test - Esperimento 3 - Variabili relative allo stato della memoria

Nome variabile	Id_Exp			
	p-value	sen	Clower	Cupper
'VmPeak'	0,0000E+00	0,0000E+00	0,0000E+00	0,0000E+00
'VmSize'	2,0204E-268	2,0392E-02	1,7631E-02	2,3638E-02
'VmHWM'	0,0000E+00	0,0000E+00	0,0000E+00	0,0000E+00
'VmRSS'	0,0000E+00	2,4085E-02	2,1344E-02	2,7102E-02
'VmPTE'	0,0000E+00	0,0000E+00	0,0000E+00	0,0000E+00
'Threads'	4,3599E-18	0,0000E+00	0,0000E+00	0,0000E+00
'MemFree'	0,0000E+00	-1,3099E-01	-1,3344E-01	-1,2837E-01
'Buffers'	0,0000E+00	3,7500E-02	3,6822E-02	3,8173E-02
'Cached'	0,0000E+00	8,5515E-03	8,5447E-03	8,5586E-03
'Active'	0,0000E+00	2,7661E-01	2,7348E-01	2,7971E-01
'Inactive'	0,0000E+00	-1,6068E-01	-1,6278E-01	-1,5831E-01
'LowFree'	0,0000E+00	-1,3099E-01	-1,3344E-01	-1,2837E-01
'Dirty'	2,4468E-01	0,0000E+00	0,0000E+00	0,0000E+00
'Writeback'	7,1433E-01	0,0000E+00	0,0000E+00	0,0000E+00
'AnonPages'	0,0000E+00	6,9621E-02	6,7558E-02	7,1605E-02
'Mapped'	0,0000E+00	0,0000E+00	0,0000E+00	0,0000E+00
'Slab'	0,0000E+00	3,4260E-03	3,3333E-03	3,5186E-03
'PageTables'	0,0000E+00	2,3419E-03	1,6310E-03	3,2672E-03
'Committed_AS'	'	0,0000E+00	1,6919E-01	1,6491E-01
'NumOfAllocFH'		7,5877E-08	0,0000E+00	0,0000E+00

Tabella 4.12: Risultati Mann-Kendall test - Esperimento 4 - Variabili relative allo stato della memoria

Nome variabile	Id_Exp 4			
	p-value	sen	Clower	Cupper
'VmPeak'	0,0000E+00	2,6600E-01	2,6406E-01	2,6792E-01
'VmSize'	0,0000E+00	2,3698E-01	2,3390E-01	2,4010E-01
'VmHWM'	0,0000E+00	1,4913E-01	1,4779E-01	1,5050E-01
'VmRSS'	0,0000E+00	1,3536E-01	1,3333E-01	1,3739E-01
'VmPTE'	0,0000E+00	6,1444E-04	6,0469E-04	6,2451E-04
'Threads'	3,3122E-22	0,0000E+00	0,0000E+00	0,0000E+00
'MemFree'	0,0000E+00	-2,0786E+00	-2,1193E+00	-2,0382E+00
'Buffers'	0,0000E+00	4,2051E-01	4,1668E-01	4,2487E-01
'Cached'	0,0000E+00	1,4460E+00	1,3989E+00	1,4962E+00
'Active'	0,0000E+00	2,5237E+00	2,4831E+00	2,5638E+00
'Inactive'	0,0000E+00	-4,9128E-01	-4,9560E-01	-4,8698E-01
'LowFree'	0,0000E+00	-2,0786E+00	-2,1193E+00	-2,0382E+00
'Dirty'	1,7560E-11	-4,1623E-04	-5,5096E-04	-2,8551E-04
'Writeback'	4,6452E-04	0,0000E+00	0,0000E+00	0,0000E+00
'AnonPages'	0,0000E+00	1,3811E-01	1,3611E-01	1,4010E-01
'Mapped'	0,0000E+00	1,0532E-04	1,0155E-04	1,0920E-04
'Slab'	0,0000E+00	7,2570E-02	7,1424E-02	7,3748E-02
'PageTables'	0,0000E+00	6,3042E-04	6,1920E-04	6,4154E-04
'Committed_AS'	0,0000E+00	2,4483E-01	2,4130E-01	2,4840E-01
'NumOfAllocFH'	2,0525E-58	0,0000E+00	0,0000E+00	0,0000E+00

Tabella 4.13: Risultati Mann-Kendall test - Esperimento 1-2-3-4 - Variabili relative alle prestazioni del sistema

Nome variabile	Id_Exp 1			
	p-value	sen	Clower	Cupper
'Success Rate'	6,6880E-01	0,0000E+00	0,0000E+00	0,0000E+00
'Elapsed Time'	1,5197E-52	-5,6897E-06	-6,4148E-06	-4,9717E-06
'Latency Time'	4,5900E-02	9,1369E-07	1,5881E-08	1,8027E-06
Nome variabile	Id_Exp 2			
	p-value	sen	Clower	Cupper
'Success Rate'	2,0759E-128	0,0000E+00	0,0000E+00	0,0000E+00
'Elapsed Time'	1,5053E-278	2,0533E-07	1,5088E-07	3,2270E-07
'Latency Time'	6,1552E-162	6,3275E-06	4,2731E-06	9,8938E-06
Nome variabile	Id_Exp 3			
	p-value	sen	Clower	Cupper
'Success Rate'	6,9700E-01	0,0000E+00	0,0000E+00	0,0000E+00
'Elapsed Time'	7,6000E-02	-9,3699E-07	-1,9550E-06	9,5790E-08
'Latency Time'	3,2004E-168	-1,1592E-05	-1,2732E-05	-1,0499E-05
Nome variabile	Id_Exp 4			
	p-value	sen	Clower	Cupper
'Success Rate'	8,9960E-01	0,0000E+00	0,0000E+00	0,0000E+00
'Elapsed Time'	4,6878E-07	7,3330E-07	4,4960E-07	1,0183E-06
'Latency Time'	1,6073E-47	4,5352E-06	3,8463E-06	5,2910E-06

Tipo di richiesta: Heavy (In particolare in questo caso le query join).

Variazione: NO.

4.5.4 Valutazione dei trend osservati rispetto alle caratteristiche del workload

Una volta individuata l'eventuale presenza del trend aging, è importante capire se esistono delle dipendenze tra il fenomeno oggetto di studio e il workload mostrato in letteratura [13, 27, 22, 12, 11, 10, 15, 14]. Nei grafici 4.14, 4.15, 4.17 vengono rappresentati, per le variabili più significative, il trend rilevato per ciascun esperimento, al fine di osservare le differenze tra il workload il suo impatto sul fenomeno di Software Aging.

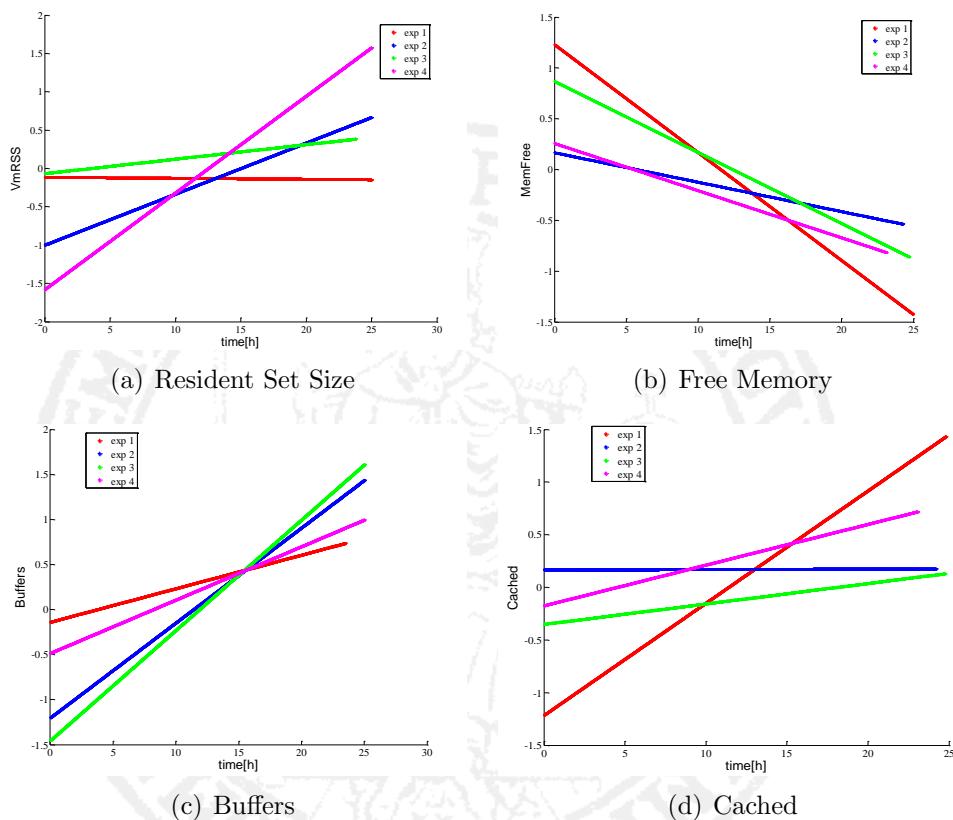


Figura 4.14: Trend rilevati per le variabili monitorate

La variabile LowFree, come già detto, in questo caso coincide con MemFree per cui il relativo trend non viene rappresentato. Viene anche rappresentato il trend della

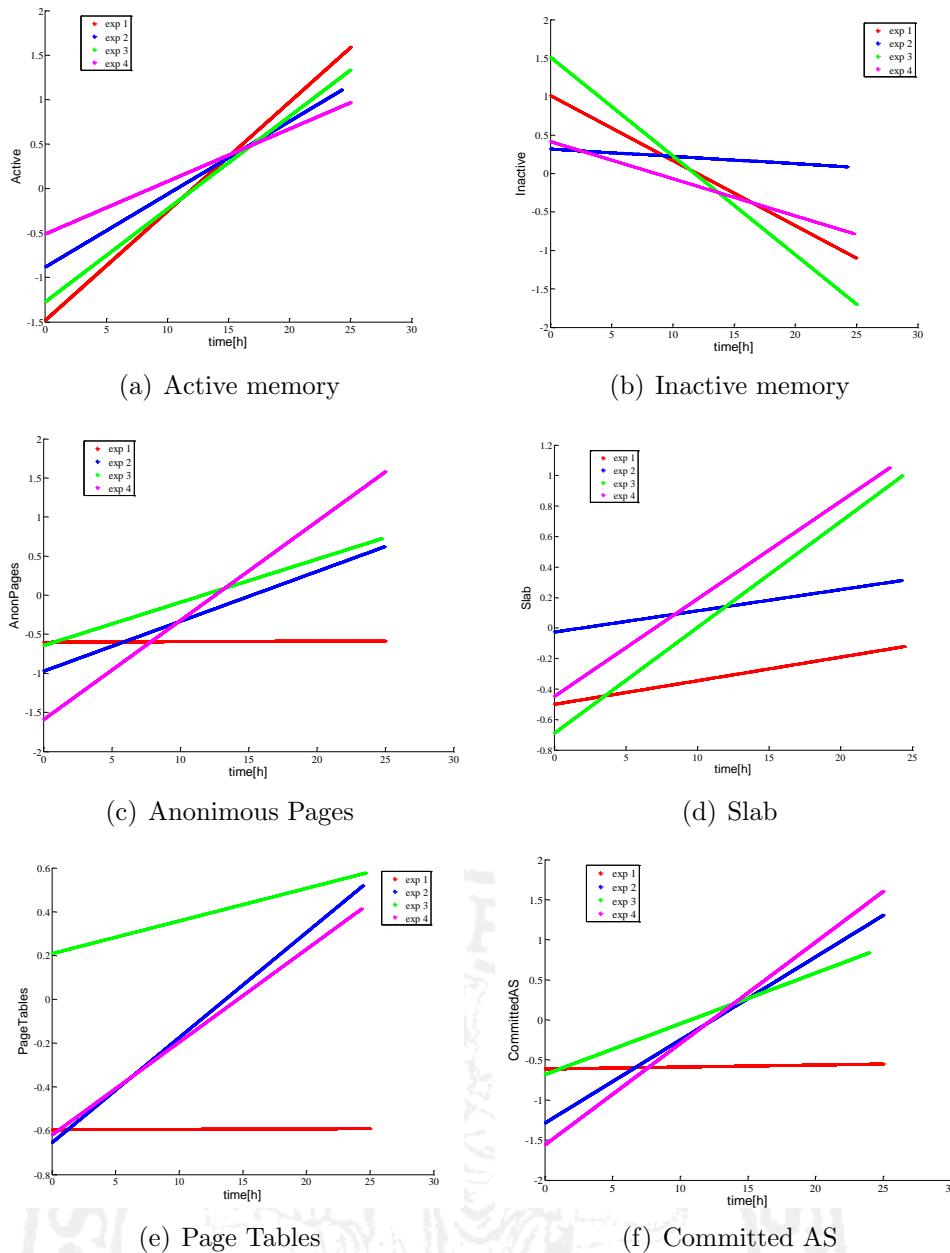


Figura 4.15: Trend rilevati per le variabili monitorate

variabile Buffers anche se per tale variabile occorre effettuare delle osservazioni particolari. La variabile Buffers, riportata in tutti gli esperimenti, indica la quantità di memoria RAM utilizzata per i buffers dei file ed ha solitamente un andamento del tipo rappresentato in 4.5.4.

La richiesta di allocazione del buffer, infatti, aumenta fino a quando non arriva alla soglia massima consentita. In questo caso il test di Mann-Kendall dà come risultato i trend rappresentati nel grafico 4.16.

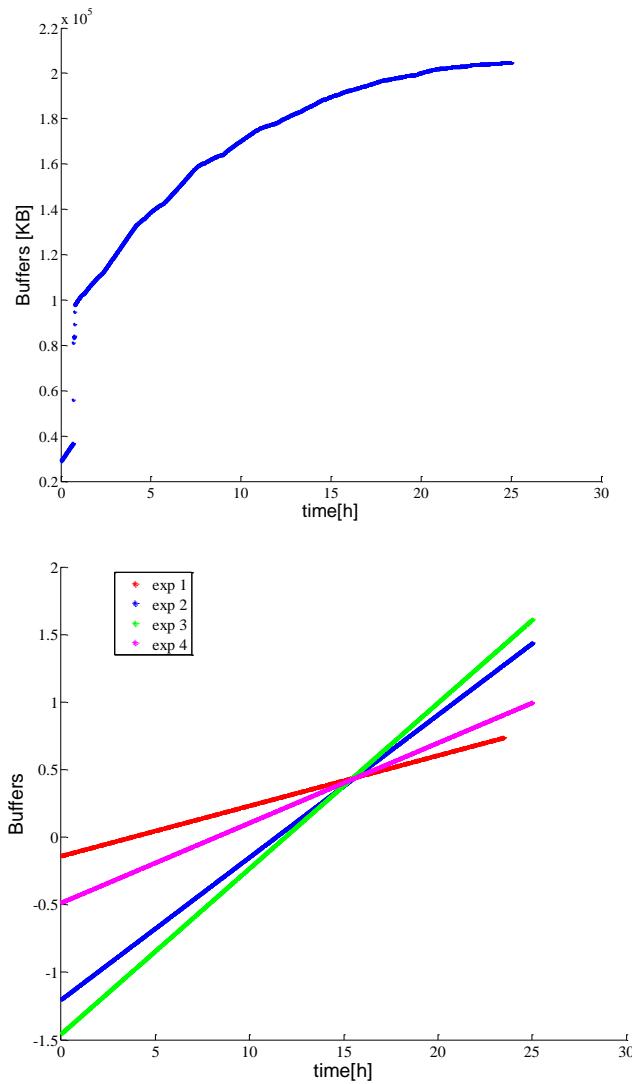


Figura 4.16: Trend della variabile Buffers

Il test, in questo caso, non risulta molto efficace poiché essendo presente un lungo tratto in cui la variabile assume valore praticamente costante la determinazione del trend lineare dà un risultato falsato. In realtà occorrerebbe determinare il trend nel primo tratto fino al raggiungimento del ginocchio.

Risulta molto interessante valutare i trend anche per le variabili indicative delle prestazioni del sistema. In particolare, la variabile SuccessRate non ha presentato trend in nessun degli esperimenti. Nel terzo esperimento non sono stati rilevati trend per nessuna delle variabili SuccessRate, Elapsed, Latency. Nel primo esperimento è stato individuato un trend soltanto per il tempo di latenza medio. Gli esperimenti 2 e 4,

invece, presentavano un trend sia per la variabile Elapsed che per Latency. I grafici 4.17 mostrano quanto appena detto.

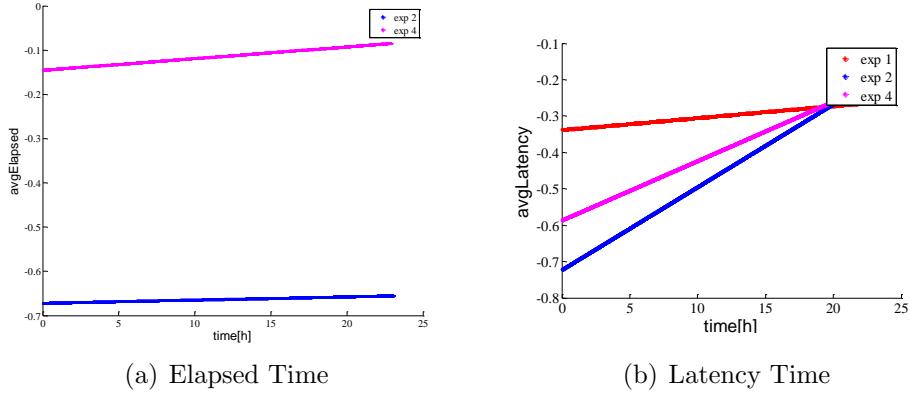


Figura 4.17: Trend rilevati per le variabili monitorate - indicatori delle prestazioni del sistema

Dai dati osservati in 4.5.3 si intuisce che il terzo esperimento, l'unico con dimensione dei dati di tipo low, è quello che stressa di meno il sistema. Gli esperimenti 2 e 4 presentano il maggior numero di variabili con trend rilevabili. Tali esperimenti differiscono dagli altri due per il fattore intensità che in quel caso è high.

I risultati, dunque, mostrano che nello studio del fenomeno del Software Aging sembra giocare un ruolo importante l'intensità e la dimensione della richiesta. Gli esperimenti 1, 2 e 4 mostrano trend più significativi, i primi due hanno dimensione high, mentre l'ultimo ha dimensione low ma intensità molto elevata.

Inoltre, i grafici 4.17 mostrano, nella maggior parte dei casi un comportamento analogo per coppie di esperimenti, in particolare:

- 4 Variabili presentano negli esperimenti 1 e 3 trend più rilevante
- 5 Variabili presentano negli esperimenti 2 e 4 trend più rilevante

Questi risultati sono di notevole importanza perché confermano quanto si trova in letteratura e possono dare un grosso contributo alla knowledge che si sta formando sul fenomeno del Software Aging, un fenomeno molto studiato ma ancora in una fase evolutiva.

Tabella 4.14: TTE (in ore (a), giorni (b)) relativi alle variabili MemFree e Active per tutti gli esperimenti

(a)			(b)		
		Nome variabili			Nome variabili
Id_Exp	'MemFree'	'Active'	Id_Exp	'MemFree'	'Active'
1	183	147	1	7	6
2	915	819	2	38	34
3	8484	2795	3	353	116
4	714	558	4	29	23

4.5.5 Stima del Time To Exaustion (TTE)

La stima del TTE è un importante aspetto nello studio del Software Aging come precedentemente accennato 1.3.1.2. Osservando i risultati riguardanti la memoria, le variabili MemFree e Active sono quelle che presentano un trend in tutti gli esperimenti, in alcuni casi molto rilevante in altri meno. Per queste due variabili e per ciascun esperimento eseguito viene calcolato il TTE secondo la regola definita in [11]. In particolare, viene utilizzata la formula lineare:

$$y = mx + c \quad (4.6)$$

dove m è la pendenza del trend rilevato, c è l'intercetta con l'asse delle ordinate della retta che identifica il trend o il valore iniziale assunto dalla variabile considerata e y il valore finale (valore minimo o massimo). Infatti, come definito in 4.5.3 il comportamento di tali variabili è stato modellato con un metodo di regressione lineare, il test di Mann-kendall.

In Tabella 4.14.(a) sono mostrati i TTE, indicati in ore, per le due variabili considerate e per tutti gli esperimenti. È immediato osservare che il terzo esperimento ha TTE molto grandi, costituisce quindi, a conferma di quanto detto in 4.5.3, l'esperimento che stressa di meno il sistema. La Tabella 4.14.(b) riporta gli stessi valori di TTE calcolati in giorni.

Non sopprimer mai un'idea,
trasformala.

3M Company

Capitolo 5.

Conclusioni e Sviluppi Futuri

Le problematiche di affidabilità dei sistemi software, relativamente al fenomeno del software aging, sono state illustrate nel primo capitolo. La maggior parte dei lavori, il cui obiettivo è indagare sulla dipendenza tra fenomeno oggetto di studio e il workload, fornisce delle soluzioni *non generali*, perché legate alla specifica applicazione o perché riguardanti parametri del sistema di basso livello. A partire da queste considerazioni, nasce l'esigenza di astrarre' il concetto di workload, in modo da poter applicare le soluzioni trovate a *qualsiasi* tipo di sistema e renderne possibile la comparazione, in termini di 'sofferenza di aging', una sorta di '*aging benchmarking*'. Il metodo è basato su un processo di caratterizzazione del workload che, partendo da una descrizione ad alto livello, conduce alla progettazione di un insieme di esperimenti da eseguire su una o più applicazioni da esaminare. Il piano sperimentale si concentra particolarmente sulla scelta della tipologia di progettazione rispetto al fattore software. E' importante sottolineare che il diverso ruolo dei fattori coinvolti nell'esperimento conduce a scelte strategiche diverse nella pianificazione e cioè a diversi disegni sperimentali e quindi scegliere erroneamente di adottare un disegno piuttosto che un altro, potrebbe portare a risultati *non rappresentativi* del fenomeno. Poiché il software rispecchia la condizione in cui si svolge un l'esperimento, rappresenta un fattore sub-sperimentale (o di disturbo). Il caso di studio analizzato ha riguardato il DBMS MySQL, un DBMS open-source tra i più utilizzati. I risultati sperimentali ottenuti hanno dimostrato la presenza di Software Aging. In particolare, stata evidenziata una dipendenza tra il workload ed il Software Aging, la quale non può essere trascurata nella realizzazione di modelli di previsione.

che stimano la tempificazione per le operazioni di Software di Rejuvenation. Infatti, conoscendo l'influenza del workload sulle dinamiche di aging, si pu stimare il Time To Exaustion in base al workload sottoposto al sistema in un certo periodo. In questo modo possibile conoscere come il sistema si comporterà successivamente, evitando così di monitorare continuamente le risorse del sistema stesso. Ad esempio, nel caso di un web server che gestisce dati sensibili, possibile dirottare le richieste ad una standby spare o ad un hot spare, provvedendo al riavvio del server principale quando necessario. Molti studi, il cui scopo era quello di studiare il fenomeno del Software Aging rispetto al suo legame con il workload focalizzavano la loro attenzione su particolari variabili, come la memoria libera e il throughput. Attraverso un'attenta analisi, questo lavoro ha mostrato che è opportuno monitorare anche altre variabili, indicative della salute del sistema, elencate in 5.1.

Tabella 5.1: Variabili da monitorare

<i>VmSize</i>
<i>VmRSS</i>
<i>VmPTE</i>
<i>Threads</i>
<i>MemFree</i>
<i>Buffers</i>
<i>Cached</i>
<i>Active</i>
<i>Inactive</i>
<i>LowFree</i>
<i>Dirty</i>
<i>Writeback</i>
<i>AnonPages</i>
<i>Mapped</i>
<i>Slab</i>
<i>PageTables</i>

Il contributo informativo delle altre variabili considerate risulta infatti molto rilevante. Inoltre non è sempre detto che il comportamento di tali variabili sia ben modellato con metodi di regressione lineare. Lo studio e la metodologia proposta servono ad identificare le variabili più significative da inserire nei modelli di stima del TTE ed in generale il TTF (Time To Failure), sia ad alto livello (intensità, dimensione, tipo di richiesta, variazione) che a basso livello (threads, VmRss, MemFree, VmSize, ecc...).

Appendice A.

Struttura e Script per il Database TPC-E

Questa appendice mostra la struttura del database TPC-E utilizzato nel contesto del presente lavoro di tesi. Solo le tabelle, le relazioni, e gli script utilizzate verranno mostrate di seguito.

A.1 Schema ER TPC-E

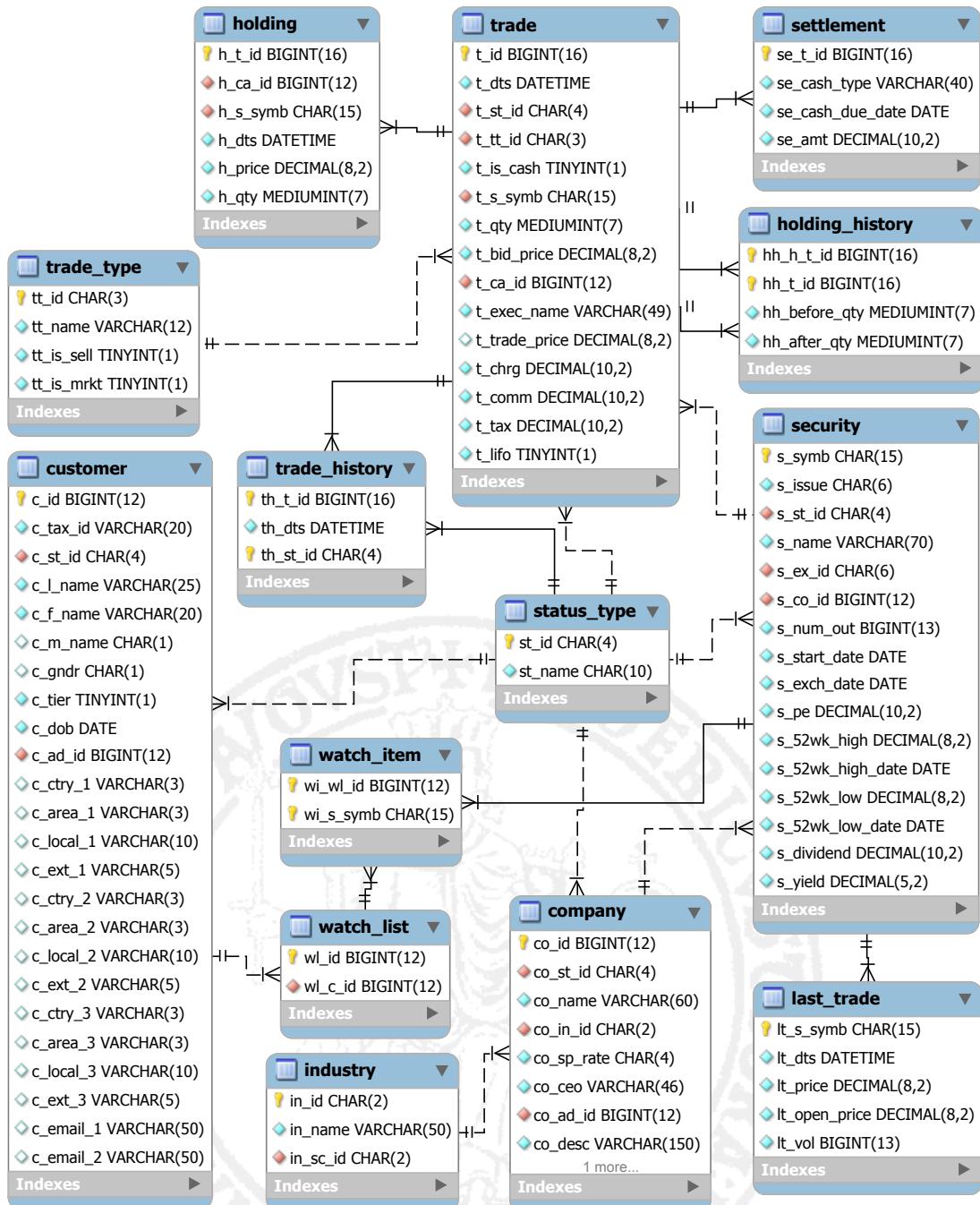


Figura A.1: Schema ER delle Tabelle utilizzate in TPC-E

A.2 Script per il caricamento del database

```

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
LOAD DATA LOCAL INFILE "../flat_out/AccountPermission.txt"
    INTO TABLE account_permission FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/Customer.txt"
    INTO TABLE customer FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/CustomerAccount.txt"
    INTO TABLE customer_account FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/CustomerTaxrate.txt"
    INTO TABLE customer_taxrate FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/Holding.txt"
    INTO TABLE holding FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/HoldingHistory.txt"
    INTO TABLE holding_history FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/HoldingSummary.txt"
    INTO TABLE holding_summary FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/WatchItem.txt"
    INTO TABLE watch_item FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/WatchList.txt"
    INTO TABLE watch_list FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/Broker.txt"
    INTO TABLE broker FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/CashTransaction.txt"
    INTO TABLE cash_transaction FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/Charge.txt"
    INTO TABLE charge FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/CommissionRate.txt"
    INTO TABLE commission_rate FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/Settlement.txt"
    INTO TABLE settlement FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/Trade.txt"
    INTO TABLE trade FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/TradeHistory.txt"
    INTO TABLE trade_history FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/TradeRequest.txt"
    INTO TABLE trade_request FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/TradeType.txt"
    INTO TABLE trade_type FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/Company.txt"
    INTO TABLE company FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/CompanyCompetitor.txt"
    INTO TABLE company_competitor FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/DailyMarket.txt"
    INTO TABLE daily_market FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/Exchange.txt"
    INTO TABLE exchange FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/Financial.txt"
    INTO TABLE financial FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/Industry.txt"
    INTO TABLE industry FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/LastTrade.txt"
    INTO TABLE last_trade FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/NewsItem.txt"
    INTO TABLE news_item FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/NewsXRef.txt"
    INTO TABLE news_xref FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/Sector.txt"
    INTO TABLE sector FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/Security.txt"
    INTO TABLE security FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/Address.txt"
    INTO TABLE address FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/StatusType.txt"
    INTO TABLE status_type FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/Taxrate.txt"
    INTO TABLE taxrate FIELDS TERMINATED BY "|";
LOAD DATA LOCAL INFILE "../flat_out/ZipCode.txt"
    INTO TABLE zip_code FIELDS TERMINATED BY "|";
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;

```

```
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

A.3 Script per la creazione degli indici del database

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
CREATE INDEX i_c_tax_id ON customer (c_tax_id);
CREATE INDEX i_co_name ON company (co_name);
CREATE INDEX i_th_t_id_dts ON trade_history (th_t_id , th_dts);
CREATE INDEX i_in_name ON industry (in_name);
CREATE INDEX i_sc_name ON sector (sc_name);
CREATE INDEX i_t_ca_id_dts ON trade (t_ca_id , t_dts);
CREATE INDEX i_t_s_symb_dts ON trade (t_s_symb , t_dts);
CREATE INDEX i_h_ca_id_s_symb_dts ON holding (h_ca_id , h_s_symb , h_dts);
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

A.4 Script per la creazione delle chiavi

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
ALTER TABLE account_permission ADD CONSTRAINT fk_account_permission_ca
  FOREIGN KEY (ap_ca_id) REFERENCES customer_account (ca_id);
ALTER TABLE customer ADD CONSTRAINT fk_customer_st
  FOREIGN KEY (c_st_id) REFERENCES status_type (st_id);
ALTER TABLE customer ADD CONSTRAINT fk_customer_ad
  FOREIGN KEY (c_ad_id) REFERENCES address (ad_id);
ALTER TABLE customer_account ADD CONSTRAINT fk_customer_account_b
  FOREIGN KEY (ca_b_id) REFERENCES broker (b_id);
ALTER TABLE customer_account ADD CONSTRAINT fk_customer_account_c
  FOREIGN KEY (ca_c_id) REFERENCES customer (c_id);
ALTER TABLE customer_taxrate ADD CONSTRAINT fk_customer_taxrate_tx
  FOREIGN KEY (cx_tx_id) REFERENCES taxrate (tx_id);
ALTER TABLE customer_taxrate ADD CONSTRAINT fk_customer_taxrate_c
  FOREIGN KEY (cx_c_id) REFERENCES customer (c_id);
ALTER TABLE holding ADD CONSTRAINT fk_holding_t
  FOREIGN KEY (h_t_id) REFERENCES trade (t_id);
ALTER TABLE holding ADD CONSTRAINT fk_holding_hs
  FOREIGN KEY (h_ca_id , h_s_symb) REFERENCES holding_summary (hs_ca_id , hs_s_symb);
ALTER TABLE holding_history ADD CONSTRAINT fk_holding_history_t1
  FOREIGN KEY (hh_h_t_id) REFERENCES trade (t_id);
ALTER TABLE holding_history ADD CONSTRAINT fk_holding_history_t2
  FOREIGN KEY (hh_t_id) REFERENCES trade (t_id);
ALTER TABLE holding_summary ADD CONSTRAINT fk_holding_summary_ca
  FOREIGN KEY (hs_ca_id) REFERENCES customer_account (ca_id);
ALTER TABLE holding_summary ADD CONSTRAINT fk_holding_summary_s
  FOREIGN KEY (hs_s_symb) REFERENCES security (s_symb);
ALTER TABLE watch_item ADD CONSTRAINT fk_watch_item_wl
  FOREIGN KEY (wi_wl_id) REFERENCES watch_list (wl_id);
ALTER TABLE watch_item ADD CONSTRAINT fk_watch_item_s
  FOREIGN KEY (wi_s_symb) REFERENCES security (s_symb);
ALTER TABLE watch_list ADD CONSTRAINT fk_watch_list_c
  FOREIGN KEY (wl_c_id) REFERENCES customer (c_id);
ALTER TABLE broker ADD CONSTRAINT fk_broker_st
  FOREIGN KEY (b_st_id) REFERENCES status_type (st_id);
ALTER TABLE cash_transaction ADD CONSTRAINT fk_cash_transaction_t
  FOREIGN KEY (ct_t_id) REFERENCES trade (t_id);
ALTER TABLE charge ADD CONSTRAINT fk_charge_tt
  FOREIGN KEY (ch_tt_id) REFERENCES trade_type (tt_id);
ALTER TABLE commission_rate ADD CONSTRAINT fk_commission_rate_tt
  FOREIGN KEY (cr_tt_id) REFERENCES trade_type (tt_id);
ALTER TABLE commission_rate ADD CONSTRAINT fk_commission_rate_ex
  FOREIGN KEY (cr_ex_id) REFERENCES exchange (ex_id);
ALTER TABLE settlement ADD CONSTRAINT fk_settlement_t
  FOREIGN KEY (se_t_id) REFERENCES trade (t_id);
```

```

ALTER TABLE trade ADD CONSTRAINT fk_trade_st
  FOREIGN KEY (t_st_id) REFERENCES status_type (st_id);
ALTER TABLE trade ADD CONSTRAINT fk_trade_tt
  FOREIGN KEY (t_tt_id) REFERENCES trade_type (tt_id);
ALTER TABLE trade ADD CONSTRAINT fk_trade_s
  FOREIGN KEY (t_s_symb) REFERENCES security (s_symb);
ALTER TABLE trade ADD CONSTRAINT fk_trade_ca
  FOREIGN KEY (t_ca_id) REFERENCES customer_account (ca_id);
ALTER TABLE trade_history ADD CONSTRAINT fk_trade_history_t
  FOREIGN KEY (th_t_id) REFERENCES trade (t_id);
ALTER TABLE trade_history ADD CONSTRAINT fk_trade_history_st
  FOREIGN KEY (th_st_id) REFERENCES status_type (st_id);
ALTER TABLE trade_request ADD CONSTRAINT fk_trade_request_t
  FOREIGN KEY (tr_t_id) REFERENCES trade (t_id);
ALTER TABLE trade_request ADD CONSTRAINT fk_trade_request_tt
  FOREIGN KEY (tr_tt_id) REFERENCES trade_type (tt_id);
ALTER TABLE trade_request ADD CONSTRAINT fk_trade_request_s
  FOREIGN KEY (tr_s_symb) REFERENCES security (s_symb);
ALTER TABLE trade_request ADD CONSTRAINT fk_trade_request_b
  FOREIGN KEY (tr_b_id) REFERENCES broker (b_id);
ALTER TABLE company ADD CONSTRAINT fk_company_st
  FOREIGN KEY (co_st_id) REFERENCES status_type (st_id);
ALTER TABLE company ADD CONSTRAINT fk_company_in
  FOREIGN KEY (co_in_id) REFERENCES industry (in_id);
ALTER TABLE company ADD CONSTRAINT fk_company_ad
  FOREIGN KEY (co_ad_id) REFERENCES address (ad_id);
ALTER TABLE company_competitor ADD CONSTRAINT fk_company_competitor_co1
  FOREIGN KEY (cp_co_id) REFERENCES company (co_id);
ALTER TABLE company_competitor ADD CONSTRAINT fk_company_competitor_co2
  FOREIGN KEY (cp_comp_co_id) REFERENCES company (co_id);
ALTER TABLE company_competitor ADD CONSTRAINT fk_company_competitor_in
  FOREIGN KEY (cp_in_id) REFERENCES industry (in_id);
ALTER TABLE daily_market ADD CONSTRAINT fk_daily_market_s
  FOREIGN KEY (dm_s_symb) REFERENCES security (s_symb);
ALTER TABLE exchange ADD CONSTRAINT fk_exchange_ad
  FOREIGN KEY (ex_ad_id) REFERENCES address (ad_id);
ALTER TABLE financial ADD CONSTRAINT fk_financial_co
  FOREIGN KEY (fi_co_id) REFERENCES company (co_id);
ALTER TABLE industry ADD CONSTRAINT fk_industry_sc
  FOREIGN KEY (in_sc_id) REFERENCES sector (sc_id);
ALTER TABLE last_trade ADD CONSTRAINT fk_last_trade_s
  FOREIGN KEY (lt_s_symb) REFERENCES security (s_symb);
ALTER TABLE news_xref ADD CONSTRAINT fk_news_xref_ni
  FOREIGN KEY (nx_ni_id) REFERENCES news_item (ni_id);
ALTER TABLE news_xref ADD CONSTRAINT fk_news_xref_co
  FOREIGN KEY (nx_co_id) REFERENCES company (co_id);
ALTER TABLE security ADD CONSTRAINT fk_security_st
  FOREIGN KEY (s_st_id) REFERENCES status_type (st_id);
ALTER TABLE security ADD CONSTRAINT fk_security_ex
  FOREIGN KEY (s_ex_id) REFERENCES exchange (ex_id);
ALTER TABLE security ADD CONSTRAINT fk_security_co
  FOREIGN KEY (s_co_id) REFERENCES company (co_id);
ALTER TABLE address ADD CONSTRAINT fk_address_zc
  FOREIGN KEY (ad_zc_code) REFERENCES zip_code (zc_code);
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

```

Appendice B.

Query TPC-E utilizzate

Questa appendice mostra le query TPC-E utilizzate nel contesto del presente lavoro di tesi. In particolare, esse vengono classificate per tipo di richiesta e per dimensione dei dati scambiati (*low*, *high*) tra il client e il server.

B.1 Query per la size LOW

- Queries di visualizzazione (denominate **select semplici**):

```
select
    c_st_id ,c_l_name ,c_f_name ,c_m_name ,c_gndr ,c_tier ,c_dob ,c_ad_id ,
    c_ctry_1 ,c_area_1 ,c_local_1 ,c_ext_1 ,c_ctry_2 ,c_area_2 ,c_local_2 ,
    c_ext_2 ,c_ctry_3 ,c_area_3 ,c_local_3 ,c_ext_3 ,c_email_1 ,c_email_2
from
    customer
where
    C_ID < '4300000008';
select
    s_num_out ,s_name ,s_start_date
from
    security
where
    S_SYMBOL like '%AB%';
select
    SE_AMT,SE.CASH.DUE.DATE,SE.CASH.TYPE
from
    settlement
where
    SE_T_ID < '200000000000036';
```

```
select
    T_BID_PRICE , T_EXEC_NAME , T_IS_CASH , T_ID , T_TRADE_PRICE
from
    trade
where
    T_DTS >= '2005-01-03 09:00:01'
and
    T_DTS <= '2005-01-03 09:01:28'
limit 25;
```

- Queries di visualizzazioni innestate (denominate **select nested**):

```
select
    T_ID ,
```

```

T_S_SYMB,
T_QTY,
ST_NAME,
TH_DTS
from
  (select
      T_ID as ID
    from
      trade
    where
      t_ca_id > '43000000001'
    limit 10) as T,
  trade,
  trade_history,
  status_type
where
  T_ID = ID and
  TH_T_ID = T_ID and
  ST_ID = TH_ST_ID
limit 100;

```

```

select
  HH_H_T_ID , HH_T_ID , HH_BEFORE_QTY , HH_AFTER_QTY ,
  HH_H_T_ID , HH_T_ID , HH_BEFORE_QTY ,
  HH_H_T_ID , HH_T_ID , HH_BEFORE_QTY , HH_AFTER_QTY ,
  HH_H_T_ID , HH_T_ID , HH_BEFORE_QTY ,
  HH_AFTER_QTY , HH_H_T_ID , HH_T_ID , HH_BEFORE_QTY ,
  HH_AFTER_QTY , HH_H_T_ID , HH_T_ID , HH_BEFORE_QTY ,
  HH_AFTER_QTY , HH_H_T_ID , HH_T_ID ,
  HH_BEFORE_QTY , HH_AFTER_QTY , HH_H_T_ID ,
  HH_T_ID , HH_BEFORE_QTY , HH_AFTER_QTY ,
  HH_H_T_ID , HH_T_ID , HH_BEFORE_QTY ,
  HH_AFTER_QTY , HH_H_T_ID , HH_T_ID ,
  HH_BEFORE_QTY , HH_AFTER_QTY
from
  holding_history
where
  HH_H_T_ID =
  (SELECT h_t_id FROM holding where h_t_id = '200000000000035')
limit 100;

```

- o Queries di visualizzazione con congiunzione (denominate JOIN):

```

select
  S_SYMB
from
  industry , company , security
where
  IN_NAME = 'Audio & Video Equipment'
and
  CO_IN_ID = 'AV'
order by s_symb asc
limit 250;

```

```

select
  WLS_SYMB
from
  watch_item , watch_list
where
  WI_WL_ID = WL_ID
and
  WL_C_ID < '4300000372'

```

```
order by
    WLS_SYMB asc
limit 250;
```

```
select
    T_BID_PRICE , T_EXEC_NAME , T_IS_CASH , TT_IS_MRKT ,
    T_TRADE_PRICE
from
    trade , trade_type
where
    T_ID < '200000000000775'
and
    T_TT_ID = 'TMB'
ORDER BY
    T_BID_PRICE asc
limit 30;
```

- Queries di modifica (denominate **update-insert-delete**):

```
update
    last_trade
set
    LT_PRICE = 21 , LT_VOL = LT_VOL + 3 ,
    LT_DTS = '2005-01-04 09:00:00'
where
    LT_S_SYMB like '%AC%';
```

```
update
    trade
set
    T_DTS = '2005-01-03 09:02:50' ,
    T_ST_ID = 'CMPT'
where
    T_ID < '20000000000048' ;
```

```
insert into
trade
    (T_ID , T_DTS , T_ST_ID , T_TT_ID , T_IS_CASH ,
    T_S_SYMB , T_QTY , T_BID_PRICE , T_CA_ID ,
    T_EXEC_NAME , T_TRADE_PRICE , T_CHRG , T_COMM , T_TAX ,
    T_LIFO)
values
    ((2000040000010 + ((? -1 ) * 10) +i) ,
    '2005-01-03 09:02:50' , 'CMPT' , 'TLB' , '1' ,
    'NOIZ' , '200' , '27.28' , '43000003325' ,
    'Fernande Plain' , '27.26' , '5.00' , '19.63' ,
    '0.00' , '0');
```

con i che va da 1 a 10.

```
insert into
holding
    ( H_T_ID , H_CA_ID , H_S_SYMB , H_DTS , H_PRICE ,
    H_QTY )
values
    ((20000001445400 + ((? -1 ) * 15) +i) ,
    '43000008481' , 'GLT' , '2005-01-03 09:17:08' ,
    '20.43' , '800'),
```

con i che va da 1 a 10.

```
delete from
    trade
where
    T_ID >= (20000400000010 + ((? -1 )* 10) +1 )
and
    T_ID <= (20000400000010 + ((? -1 ) * 10) +11 );
```

```
delete from
    holding
where
    H_T_ID >= (200000001445400+((?-1 )*15)+1)
and
    H_T_ID <= (200000001445400+((?-1 )*15)+16);
```

B.2 Query per la size High

- Queries di visualizzazione (denominate **select semplici**):

```
select
    c_st_id ,c_l_name ,c_f_name ,c_m_name ,c_gndr ,c_tier ,
    c_dob ,c_ad_id ,c_ctry_1 ,c_area_1 ,c_local_1 ,c_ext_1 ,
    c_ctry_2 ,c_area_2 ,c_local_2 ,c_ext_2 ,c_ctry_3 ,
    c_area_3 ,c_local_3 ,c_ext_3 ,c_email_1 ,c_email_2
from
    customer
where
    C_ID < '4300000750 ';
```

```
select
    s_num_out ,s_name ,s_start_date
from security where S_SYMBOL like '%A%';
```

```
select
    SE_AMT,SE_CASH_DUE_DATE,SE_CASH_TYPE
from
    settlement
where
    SE_T_ID > '200000000000036 '
limit 3100;
```

- Queries di modifica (denominate **update-insert-delete**):

```
update
    last_trade
set
    LT_PRICE = 21 , LT_VOL = LT_VOL + 3 ,
    LT_DTS = '2005-01-04 09:00:00 ' ,
    LT_OPEN_PRICE=25
where
    lt_s_symb like '\%A\%'  

or
    lt_s_symb like '\%B\%'
```

```

or          lt_s_symb like '\%C\%'
or          lt_s_symb like '\%F\%'
or          lt_s_symb like '\%G\%';

```

```

update      trade
set          T_DTS    = '2005-01-03 09:02:50',
              T_ST_ID = 'CMPT'
where        T_ID < '200000000003800';

```

- o Queries di visualizzazione con congiunzione (denominate **join**):

```

select *
from       industry , company , security
where      IN_NAME = 'Audio & Video Equipment'
and         CO_IN_ID = 'AV'
order by   s_symb asc limit 250;

```

```

select *
from       watch_item , watch_list
where      WI_WL_ID = WL_ID
and         WL_C_ID < '4300000372'
order by   WLS_SYMB asc limit 2800;

```

```

select *
from       trade , trade_type
where      T_ID < '20000000000775'
and         T_TT_ID = 'TMB'
ORDER BY   T_BID_PRICE asc limit 700;

```

- o Queries di visualizzazioni innestate (denominate **select nested**):

```

select      T_ID , T_S_SYMB , T_QTY , ST_NAME , TH_DTS
from        (
select      T_ID as ID
              from       trade
where        t_ca_id > '43000000001'
order by    T_DTS desc limit 600) as T,
trade , trade_history , status_type
where        T_ID = ID

```

```
and          TH_T.ID = T.ID  
and          ST.ID = TH_ST.ID  
order by    TH.DTS desc;
```

- Stored procedure per i due tipi di insert:

```
DROP PROCEDURE IF EXISTS `my_insert_procedure_1`;
CREATE DEFINER='mysql_user'@'%' PROCEDURE
`my_insert_procedure_1`(`IN` t_id INT)

BEGIN
    DECLARE v INT DEFAULT 1;
    set v=1;
    while v<=900 do
        insert into
        trade (
            T_ID , T_DTS , T_ST_ID , T_TT_ID , T.IS_CASH ,
            T_SSYMB , T_QTY , T_BID_PRICE , T.CA_ID ,
            T_EXEC_NAME , T_TRADE_PRICE , T.CHRG , T.COMM ,
            T_TAX , T_LIFO )
        values
            (((20000200000010 +((t_id -1 )* 900 )+v ),
            '2005-01-03 09:02:50' , 'CMPT' , 'TLB' , '1',
            'NOIZ' , '200' , '27.28' , '43000003325' ,
            'Fernande Plain' , '27.26' , '5.00' , '19.63' ,
            '0.00' , '0'));
        set v=v+1;
    end while;
end
```

```
DROP PROCEDURE IF EXISTS 'my_insert_procedure_2'
CREATE DEFINER='mysql_user'@'\%' PROCEDURE 'my_insert_procedure_2'('IN t_id INT')
BEGIN
    DECLARE v INT DEFAULT 1;
    set v=1;
    while v<=1600 do
        insert into
            holding (

```

```

H_T_ID , H_CA_ID , H_S_SYMB , H_DTS , H_PRICE , H_QTY)
values
((200000001240400 +((t_id -1 )* 1600 )+v ), '43000008481 ',
'GLT' , '2005-01-03 09:17:08' , '20.43' , '800 ');
set v=v+1;
end while;
end

```

- Stored procedure per i due tipi di delete:

```

DROP PROCEDURE IF EXISTS 'my_delete_procedure_1'
CREATE DEFINER='mysql_user '@'%' PROCEDURE 'my_delete_procedure_1'('IN t INT')
BEGIN
DECLARE v INT DEFAULT 1;
set v=1;
while v<=900 do
delete from
trade
where
t_id = (200002000000010 +((t -1 )* 900 )+v );
set v=v+1;
end while;
end

```



Appendice C.

Significato variabili monitorate

Tabella C.1: Tabella Variabili monitorate numero 1

Nome Variabile	Descrizione
Active	Memoria che è stata utilizzata più recentemente e solitamente non reclamata, ad eccezione di casi in cui è assolutamente necessaria.
AnonPages	Pagine mappate nelle tabelle delle pagine dello spazio di utente.
Bounce	Memoria utilizzata per i dispositivi di block "bounce buffers"
Buffers	Memoria relativamente temporanea per i blocchi del disco.
Cached	Memoria cache per la lettura dei files dal disco. Non include SwapCached.
Committed_AS	Quantità di memoria attualmente allocata sul sistema.
Dirty	Memoria che è in attesa di essere scritto sul disco.
Inactive	Memoria che non è stata recentemente utilizzata. E' solitamente reclamata per altri scopi.MmMemory
IPCS	Fornisce informazioni sui servizi Inter Process Commimication per il quale il processo chiamante ha accesso in lettura.
Mapped	Files che sono stati mappati, per esempios librerie.
MemFree	Somma di LowFree e HighFree.
MemTotal	Dimensione complessiva della memoria RAM utilizzabile.
NFS_Unstable	Pagine NFS mandate al server, ma non ancora utilizzate per stabilizzare la memoria. storage

Il modulo Vmstat (Virtual Memory Statistics) riporta informazioni circa i processi, la memoria, le attività di paginazione, blocchi di I/O, traps, e l'attività della CPU.

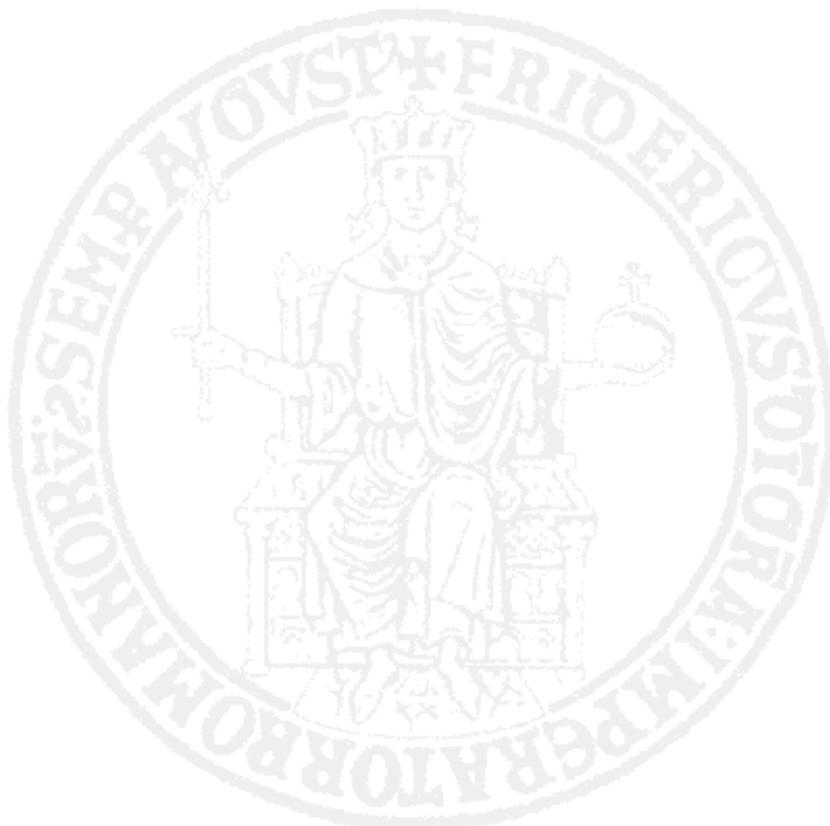
Tabella C.2: Tabella Variabili monitorate numero2

Message Queue	Una lista ordinata e strutturata di segmenti di memoria dove i processi memorizzano dati temporanei o di ritorno.
Semaphores	Fornisce un meccanismo di sincronizzazione per i processi che stanno accedendo alle stesse risorse. Nessun dato è scambiato attraverso il semaforo, quest'ultimo semplicemente coordina l'accesso alle risorse condivise.
Shared Memory	Modalità di scambio valori per i processi. Un processo potrà creare una porzione di memoria a cui gli altri processi potranno accedere.

Tabella C.3: Tabella Variabili monitorate numero 3

PageTables	Quantità di memoria dedicata al più basso livello delle tabelle delle pagine.
Slab	Memoria cache della struttura dati nel kernel.
SwapCached	Memoria che le operazioni di swap.
SwapFree	Spazio di memoria che è stato sfrattato dalla RAM, ed è temporaneamente sul disco.
SwapTotal	Quantità totale dello spazio swap disponibile.
Sysctl fs.file-nr	lista degli handles dei file nella memoria del kernel.
Threads	Numero di threads del processo.
VmallocChunk	Il più grande blocco contiguo dell'area vmalloc libero.
VmallocTotal	Dimensione complessiva dell'area di memoria vmalloc.
VmallocUsed	Quantità dell'area vmalloc usata.
VmHWM	Picco della memoria utilizzata da un particolare processo (Resident Set Size).
VmPeak	Picco della memoria virtuale.
VmPTE	Dimensioni delle entries della tabella delle pagine.
VmRSS	Memoria utilizzata da un particolare processo (Resident set size).
VmSize	Dimensione della memoria virtuale.
Writeback	Quantità di dati che si stanno trasferendo su disco.

In particolare le varibili di Vmstat considerate in questo lavoro sono riportate nella tabella seguente



Appendice D.

MySQL Traffic Generator

D.1 Class Worker

```
package Core;

import Write_risultati.write_ris;
import Write_risultati.Logger_Ris;
import log.Logger;
import log.LogObj;
import java.io.*;
import java.util.Vector;
import com.mysql.jdbc.*;
import java.net.*;

public class Worker extends Thread{

    private Espéranto e=null;
    private ThreadGenerator master;
    private int gruppo;
    private String query;
    private int indice_nel_gruppo;
    private int queryID;
    private Database db;
    //private static PrintWriter pw=null;
    private sample s=new sample();

    public Worker(ThreadGenerator t ,sample s){
        this.master=t;
        this.e=t.getEsp();
        this.query = new String();
        //this.db = dbconnector;
        this.gruppo = -1;
        this.queryID = 0;
        this.s=new sample();
    }

    public Worker(ThreadGenerator t , int g , int qID ,sample s )
    {
        this.master=t;
        this.e=t.getEsp();
        this.query = new String();
        this.db = db;
        this.gruppo = g;
        this.queryID = qID;
        this.s=new sample();
    }

    public int getGruppo()
    {
        return this.gruppo;
    }
}
```

```

public void setGruppo(int g)
{
    this.gruppo = g;
}
public void setQuery(String q)
{
    this.query = q;
}

public void run(){
    long tfinale=0;
    long tiniziale=0;
    long elapsed=0;
    long latency=0;
    write_ris oggetto_ris=new write_ris();
    Logger_Ris Logger=new Logger_Ris();
    try{

        Database db = new Database(this.master.getEsp().getHost(),
            "tpce", "mysql_user", "netmon00");
        String query_to_do=new String();
        try {
            latency=db.connect();
        } catch (ConnectException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        //System.out.println("host"+this.master.getEsp().getHost());
        this.query = master.allQueries.elementAt(gruppo-1).elementAt(queryID);
        //ESEGUE SUL DATABASE LA QUERY nella stringa query sul
        //connettore passato dal padre
        tiniziale=System.currentTimeMillis();
        this.s.setTimestampIniziale(tiniziale);
        if (this.query.indexOf("insert_1")==0)
    {
        switch (this.e.getDimensione()){
            case 'L':
                for (int i=1;i<=10;i++)
                    this.query= "insert into " +
                        "trade ( T_ID, T_DTS, T_ST_ID, T_TT_ID, T_IS_CASH,
                        T_S_SYMB, T_QTY, T_BID_PRICE, T_CA_ID, T_EXEC_NAME,
                        T_TRADE_PRICE, T_CHRG, T_COMM, T_TAX, T_LIFO) values " +
                        "((20000200000010 + ("+this.getId()+"-1 ) * 10) +" +i+" )," +
                        "'2005-01-03 09:02:50', 'CMPT', 'TLB', '1', 'NOIZ', '200', " +
                        "'27.28', '43000003325', 'Fernande Plain', '27.26', '5.00', " +
                        "'19.63', '0.00', '0');";
                    db.executeUpdate(this.query);
                }
                ; break;
            case 'H':
                for (int i=1;i<=900;i++){
                    this.query="insert into "+
                    "trade ("+
                    "T_ID, T_DTS, T_ST_ID, T_TT_ID, T_IS_CASH,"+
                    "T_S_SYMB, T_QTY, T_BID_PRICE, T_CA_ID, T_EXEC_NAME,"+
                    "T_TRADE_PRICE, T_CHRG, T_COMM, T_TAX, T_LIFO"+
                    ") "+
                    "values "+
                    "((20000200000010 +(" +this.getId()+"-1 )* 900) +" +i+" ), " +
                    "'2005-01-03 09:02:50', 'CMPT', 'TLB', '1', 'NOIZ', '200', '27.28', " +
                    "'43000003325', " +
                    "'Fernande Plain', '27.26', '5.00', '19.63', '0.00', '0');";
                    db.executeUpdate(this.query);
                }
                break;
            }
        if (this.query.indexOf("insert_2")==0)
        {
            switch (this.e.getDimensione()){

```

```

case 'L':{
    System.out.println("INS_2_low");
    for (int i=1;i<=15;i++)
        this.query= "insert into " +
            "holding ("+
            "H_T_ID,"+
            "H_CA_ID,"+
            "H_SSYMB,"+
            "H_DTS,"+
            "H_PRICE,"+
            "H_QTY"+
            ")" +
            " values " +
            "((200000001440400 +
            "((+"+this.getId()+"-1)* 15) +" +i+" ), " +
            "'43000008481', 'GLT', '2005-01-03 09:17:08 ', " +
            "'20.43', '800');";
        db.executeUpdate(this.query);
    }
    break;
}

case 'H':{
    for (int i=1;i<=1600;i++){
        this.query=" insert into " +
            "holding ("+
            "H_T_ID, H_CA_ID, H_SSYMB, H_DTS, H_PRICE, H_QTY)" +
            " values " +
            "((200000001240400 +(" +this.getId()+"-1)* 1600) +" +i+" ), " +
            "'43000008481', 'GLT', '2005-01-03 09:17:08 ', '20.43', '800');";
        db.executeUpdate(this.query);
    }
    break;
}
}

if (this.query.contains("delete_1")){
    switch (this.e.getDimensione()){

        case 'L':{
            this.query= "delete from trade where "+
                "T_ID >= (20000200000010 + ((+"+this.getId()+"-1 ) * 10) +1 ) " +
                "and T_ID <= (20000200000010 + ((+"+this.getId()+" -1 )* 10) +11 );";
            db.executeUpdate(this.query);
        }
        break;
    }
}

case 'H':{
    for (int i=1;i<=900;i++){
        this.query="delete from trade where "+
            " t_id = (20000200000010 +(" +this.getId()+" -1 )* " +
            "900) +" +i+" );";
        db.executeUpdate(this.query);
    }
    break;
}
}

if (this.query.contains("delete_2")){
    switch (this.e.getDimensione()){
        case 'L':{
            this.query= "delete from holding where "+
                "H_T_ID >= (200000001440400 + ((+"+this.getId()+" -1 ) *" +
                " 15) +1 ) and H_T_ID <= (200000001440400 +" +
                " ((+"+this.getId()+" -1 ) * 15) +16 );";
        }
    }
}

```

```

        db.executeUpdate(this.query);
    }
    ; break;
case 'H':{
    for (int i=1;i<1600;i++){
        this.query="delete from holding where "+
            "h_t_id = (200000001240400 +(" +this.getId() + " -1 )* 1600) +" +i+");";
        db.executeUpdate(this.query);
    }
    break;
}

if(this.query.contains("update ") || this.query.contains("drop "))
|| this.query.contains("create "))
{
    """+this.queryID +" query: "+this.query);
    db.executeUpdate(this.query);
if (this.query.contains("select"))
{
    Vector v = db.executeQuery(this.query);
}
+System.currentTimeMillis());
tfinale=System.currentTimeMillis();
db.disconnect();
this.s.setTimestampFinale(tfinale);
this.master.incrementTot();
elapsed=tfinale-tiniziale;
this.s.setElapsed(elapsed);
oggetto_ris.setTs(tfinale);
oggetto_ris.setElapsed(elapsed);
oggetto_ris.setLatency(latency);
oggetto_ris.setSuccess("true");
Logger_Ris.scriviOperazione(oggetto_ris);
} catch (Exception e1)
{
    e1.printStackTrace();
    long ts=System.currentTimeMillis();
    System.out.println("Errore nella query "+query +" del gruppo " +gruppo);

    oggetto_ris.setTs(tfinale);
    oggetto_ris.setElapsed(elapsed);
    oggetto_ris.setLatency(latency);
    oggetto_ris.setSuccess("false");
    Logger_Ris.scriviOperazione(oggetto_ris);
}
}

public void setQueryID(int q)
{
    this.queryID = q;
}
public String toString(){
    return this.getName()+" Gruppo="+gruppo +",queryID="+queryID;
}

private void Query_type1(int i) {
    try {
        char indice= (char) i;
        LeggiFile(1,indice);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

private void Query_type2(int i){

```

```

try {
    char indice= (char) i;
    LeggiFile(2,indice);
} catch (IOException e) {
    e.printStackTrace();
}
}

private void Query_type3(int i){
try {
    char indice= (char) i;
    LeggiFile(3,indice);
} catch (IOException e) {
// TODO Auto-generated catch block
    e.printStackTrace();
}
}

private void Query_type4(int i){
try {
    char indice= (char) i;
    LeggiFile(4,indice);
} catch (IOException e) {
// TODO Auto-generated catch block
    e.printStackTrace();
}
}

public void LeggiFile (int gruppo,char indice) throws IOException
{ FileReader fr = new FileReader("query_type"+gruppo+".txt");
BufferedReader file = new BufferedReader(fr); String a="";
while (a.compareTo("@")!=0) { a=file.readLine(); } }
}

```

D.2 Class Thread Generator

```

package Core;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.sql.Time;
import java.util.Random;
import java.util.Vector;
import java.util.concurrent.atomic.AtomicInteger;

public class ThreadGenerator extends Thread{
private Esperimento e=null;
private Decisor dec=null;
private final int TIMEUNIT=1000;           // unita di tempo in msec
private final int THREADSCALEFACTOR = 5;
private final int NORM_P=(Thread.MAX_PRIORITY+Thread.MIN_PRIORITY)/2;
private final int HIGH_P=(Thread.MAX_PRIORITY-1);
private final int LOW_P=(Thread.MIN_PRIORITY+1);
private int state;                      // 0=query gruppo 0
                                         // 1=query gruppo 1
                                         // 2=query gruppo 2
                                         // 3=query gruppo 3

private double invPeriod;
// periodo di generazione delle richieste
private AtomicInteger tot;
int uno,due,tre,quattro;
Vector<String> query_1;
}

```

```

Vector<String> query_2;
Vector<String> query_3;
Vector<String> query_4;
Vector<Vector<String>> allQueries;
Vector<query_manager> qid_vector;
static PrintWriter pw=null;
public ThreadGenerator (Esperimento exp){
    this.e=exp;
    this.invPeriod=(double)(TIME_UNIT);
    this.state=-1;
    this.dec=new Decisor(e);
    uno=due=tre=quattro=0;
    tot=new AtomicInteger(0);
    query_1 = new Vector<String>();
    query_2 = new Vector<String>();
    query_3 = new Vector<String>();
    query_4 = new Vector<String>();
    allQueries = new Vector<Vector<String>>();
    qid_vector = new Vector<query_manager>();
    /*
     * ogni gruppo ha un id manager
     * quindi 4 gruppi -> 4 oggetti query id manager
     * sono organizzati in un vettore
     * così quando si sceglie il gruppo ad esempio da getDecision
     * si usa il valore del "gruppo" come entry per selezionare
     * l'oggetto query id manager su cui invocare il getQueryID ();
     */
}
public void run() {
    /*
     * Thread generator legge tutte le query da tutti i
     * files e le memorizza in una matrice di stringhe
     * la matrice è organizzata così:
     * QUERY(X,INDICE) -> query INDICE del gruppo X
     * In questo modo i thread worker non dovranno accedere
     * al file volta per volta e minimizzeranno il loro tempo
     * di esecuzione
     * inoltre in questo modo al thread worker verrà passato
     * l'elemento della matrice QUERY(GRUPPO, INDICE) e i thread
     * non dovranno fare altro che eseguire la query
     * contenuta nella variabile passata
     */
    int query_id;
    String query_string=null;
    Vector<String> query = new Vector<String>();
    Vector<String> query_iniziali = new Vector<String>();
    Vector <String>[] queries;
    Vector<String> query_appoggio=new Vector<String>();
    FileInputStream fstream = null;
    switch (this.e.getTiporichiesta()) {
    case 'L': {
        try {
            fstream = new FileInputStream("query_LIGHT.txt");
        } catch (FileNotFoundException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
        DataInputStream in = new DataInputStream(fstream);
        BufferedReader br = new BufferedReader(new InputStreamReader(in));
        String strLine;
        try {
            while ((strLine = br.readLine()) != null)
                query_iniziali.add(strLine);
        } catch (IOException e1) {
            e1.printStackTrace();
        }
    }
    switch (this.e.getDimensione()){
    case 'L':{

```

```

        for (int i=0;i<query_iniziali.size();i++){
            query_string=query_iniziali.elementAt(i)+"_L.txt";
            System.out.print("file : "+query_iniziali.elementAt(i)+"_L.txt");
            query.add(i, query_string);
        }
    }; break;
case 'H':{
    for (int i=0;i<query_iniziali.size();i++){
        query_string=query_iniziali.elementAt(i)+"_H.txt";
        query.add(i, query_string);
    }
}; break;
break;
case 'H': try {
    fstream = new FileInputStream("query_HEAVY.txt");
} catch (FileNotFoundException e1) {
    // TODO Auto-generated catch block
    e1.printStackTrace();
}
DataInputStream in = new DataInputStream(fstream);
BufferedReader br = new BufferedReader(new InputStreamReader(in));
String strLine;
try {
    while ((strLine = br.readLine()) != null)
        query_iniziali.add(strLine);
} catch (IOException e1) {
    // TODO Auto-generated catch block
    e1.printStackTrace();
}
switch (this.e.getDimensione()){
case 'L':{
    for (int i=0;i<query_iniziali.size();i++){
        query_string=query_iniziali.elementAt(i)+"_L.txt";
        System.out.print("file : "+query_iniziali.elementAt(i)+"_L.txt");
        query.add(i, query_string);
    }
}; break;
case 'H':{
    for (int i=0;i<query_iniziali.size();i++){
        query_string=query_iniziali.elementAt(i)+"_H.txt";
        query.add(i, query_string);
    }
}; break;
}
System.out.print("numero gruppo query che hai richiesto: "+query.size());
for (int i=0;i<query.size();i++){
try {
    System.out.print("elem: "+query.elementAt(i));
    //l'array di Vector "queries" contiene per ogni Vector le query di
    //un particolare gruppo
    fstream = new FileInputStream(query.elementAt(i));
    DataInputStream in = new DataInputStream(fstream);
    BufferedReader br = new BufferedReader(new InputStreamReader(in));
    String strLine;

    while ((strLine = br.readLine()) != null)
        query_appoggio.add(strLine);
    //allQueries.add(query_1);
    //queries[i].add(strLine);
    //query_1.add(strLine);
    allQueries.add(query_appoggio);
} catch (FileNotFoundException e2) {
    // TODO Auto-generated catch block
    e2.printStackTrace();
    System.err.println("FILE query not found... aborting");
} catch (IOException e1) {
    // TODO Auto-generated catch block
    e1.printStackTrace();
}
}
}

```

```

for (int i=0; i<query.size(); i++)
{
    qid_vector.add(new query_manager(i+1,0, allQueries.elementAt(i).size()));

}

long t1=0,t2=0,end,durataMSec,endgruppo;
durataMSec=(long)(this.e.getDurata()*1000);
long startTime = System.currentTimeMillis();
end=durataMSec+startTime;
int gruppo=1;
//istanzia il connettore al database che passa ai thread creati
sample s=new sample();
if (this.e.getVariazione() == 0)
//non c'è bisogno di decisioni esecuzioni per ciascun gruppo di query per
//la durata:durata_gruppo
{
    System.out.println("start experiment NO VARIATION... hold on");
    Worker[] th1 = new Worker[e.getIntensita()*THREADSCALEFACTOR];
    for(int j=0;j<THREADSCALEFACTOR*e.getIntensita();j++)
    {
        th1[j] = new Worker(this,s);
        th1[j].setPriority(this.NORMP);
    }
    while(gruppo<=allQueries.size())
    {
        int initIndex = 0;
        int intensity = e.getIntensita();
        for(int j=0;j<THREADSCALEFACTOR*e.getIntensita();j++)
        {
            th1[j].setGruppo(gruppo);
            th1[j].setQueryID(qid_vector.elementAt(gruppo-1).getquery_id());
        }
        startTime = System.currentTimeMillis();
        while(System.currentTimeMillis()-startTime <durataMSec)
            while(System.currentTimeMillis()-startTime<durataMSec/allQueries.size())
            {
                int j=0;
                for( j=initIndex;j<initIndex+intensity;j++)
                {
                    try{
                        th1[j%(THREADSCALEFACTOR*intensity)].start();
                    }catch(IllegalThreadStateException e)
                    {
                        th1[j%(THREADSCALEFACTOR*intensity)] =
                        new Worker(this,gruppo,qid_vector.elementAt(gruppo-1).getquery_id(),s );
                        th1[j%(THREADSCALEFACTOR*intensity)].start();
                    }
                }
                initIndex = j;
                try {
                    Thread.sleep(TIME_UNIT);
                } catch (InterruptedException e1) {
                    e1.printStackTrace();
                }
                //dorme per un tempo pari al tempo di ciclo - 1
                //secondo meno il tempo necessario a creare gli e.getIntensita Th
                //end while systemtime<durata
                gruppo++;
            }
        }//while gruppo<4
    }
    else if (this.e.getVariazione() > 0)
    {
        System.out.println("start experiment VARIATION... hold on");
        Worker[] th1 = new Worker[e.getIntensita()*THREADSCALEFACTOR];
        for(int j=0;j<e.getIntensita()*THREADSCALEFACTOR;j++)
        {
            th1[j] = new Worker(this,s);
            th1[j].setPriority(this.NORMP);
        }
        int initIndex = 0;
        int intensity = e.getIntensita();
        startTime = System.currentTimeMillis();
        end=durataMSec+startTime;
    }
}

```

```

while(System.currentTimeMillis()<end)
{
    int j;
    for( j=initIndex ;j<initIndex+intensity ;j++)
    {
        gruppo = dec.getDecision()%allQueries.size()+1;
        query_id= qid_vector.elementAt(gruppo-1).getquery_id();
        th1[j%(THREADSCALEFACTOR*intensity)].setQueryID(query_id);
        //TOGLIERLA QUANDO E' finito il debug
        th1[j%(THREADSCALEFACTOR*intensity)].setGruppo(gruppo);
        try{
            th1[j%(THREADSCALEFACTOR*intensity)].start();
        }catch(IllegalThreadStateException e)
        {
            th1[j%(THREADSCALEFACTOR*intensity)] =
            new Worker(this,gruppo,query_id,s );
            th1[j%(THREADSCALEFACTOR*intensity)].setPriority(this.NORMP);
            th1[j%(THREADSCALEFACTOR*intensity)].start();
        }
        switch(gruppo)
        {
            case 1:
                uno++;
                break;
            case 2:
                due++;
                break;
            case 3:
                tre++;
                break;
            case 4:
                quattro++;
                break;
        }
    }
    initIndex = j;
    try {
        Thread.sleep(TIME_UNIT);
    } catch (Exception e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
        System.out.println("time unit: " +TIME_UNIT*System.currentTimeMillis());
    }
    //dorme per un tempo pari al tempo di ciclo - 1
    //secondo meno il tempo necessario a creare gli e.getIntensita Th
    //end while System.currentTimeMillis()<end
} //end if getvariazione >0
    System.out.println("Thread generator terminato!\n");
}
public Esperimento getEsp(){
    return this.e;
}
public void incrementTot(){
    this.tot.incrementAndGet();
}
public Decisor getDec() {
    return dec;
}
public int getUno() {
    return uno;
}
public int getDue() {
    return due;
}
public int getTre() {
    return tre;
}
public int getQuattro() {
    return quattro;
}
public int getTot() {
    return tot.get();
}

```

}



Bibliografia

- [1] Jim Gray and Daniel P. Siewiorek. High-availability computer systems. *Computer*, 24:39–48, September 1991.
- [2] J. Gray. A census of tandem system availability between 1985 and 1990. *Reliability, IEEE Transactions on*, 39(4):409 –418, oct 1990.
- [3] E. Voas, F. Charron, G. McGraw, K. Miller, and M. Friedman. Predicting how badly ldquo;good rdquo; software can behave. *Software, IEEE*, 14(4):73 –83, jul/aug 1997.
- [4] M. Grottke, R. Matias, and K.S. Trivedi. The fundamentals of software aging. In *Software Reliability Engineering Workshops, 2008. ISSRE Wksp 2008. IEEE International Conference on*, pages 1 –6, nov. 2008.
- [5] S. Garg, A. van Moorsel, K. Vaidyanathan, and K.S. Trivedi. A methodology for detection and estimation of software aging. In *Software Reliability Engineering, 1998. Proceedings. The Ninth International Symposium on*, pages 283 –292, nov 1998.
- [6] Y. Huang, C. Kintala, N. Kolettis, and N.D. Fulton. Software rejuvenation: analysis, module and applications. In *Fault-Tolerant Computing, 1995. FTCS-25. Digest of Papers., Twenty-Fifth International Symposium on*, pages 381 –390, jun 1995.
- [7] A. Avritzer, A. Bondi, M. Grottket, K.S. Trivedi, and E.J. Weyuker. Performance assurance via software rejuvenation: Monitoring, statistics and algorithms. In *Dependable Systems and Networks, 2006. DSN 2006. International Conference on*, pages 435 –444, june 2006.
- [8] T. Dohi, K. Goseva-Popstojanova, and K.S. Trivedi. Statistical non-parametric algorithms to estimate the optimal software rejuvenation schedule. In *Dependable Computing, 2000. Proceedings. 2000 Pacific Rim International Symposium on*, pages 77 –84, 2000.
- [9] K. Vaidyanathan and K.S. Trivedi. A comprehensive model for software rejuvenation. *Dependable and Secure Computing, IEEE Transactions on*, 2(2):124 – 137, april-june 2005.
- [10] K. Vaidyanathan and K.S. Trivedi. A measurement-based model for estimation of resource exhaustion in operational software systems. In *Software Reliability*

Engineering, 1999. Proceedings. 10th International Symposium on, pages 84 –93, 1999.

- [11] R. Matias and P.J.F. Filho. An experimental study on software aging and rejuvenation in web servers. In *Computer Software and Applications Conference, 2006. COMPSAC '06. 30th Annual International*, volume 1, pages 189 –196, sept. 2006.
- [12] Antonio Bovenzi, Domenico Cotroneo, Roberto Pietrantuono, and Stefano Russo. Workload characterization for software aging analysis. In *To appear In proc. of the International Symposium on Software Reliability Engineering (ISSRE) 2011*, New York, NY, USA, 2011. IEEE.
- [13] L. Silva, H. Madeira, and J.G. Silva. Software aging and rejuvenation in a soap-based server. In *Network Computing and Applications, 2006. NCA 2006. Fifth IEEE International Symposium on*, pages 56 –65, july 2006.
- [14] R. Matias and P.J.F. Filho. An experimental study on software aging and rejuvenation in web servers. In *Computer Software and Applications Conference, 2006. COMPSAC '06. 30th Annual International*, volume 1, pages 189 –196, sept. 2006.
- [15] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *Dependable and Secure Computing, IEEE Transactions on*, 1(1):11 – 33, jan.-march 2004.
- [16] R. Chillarege, I.S. Bhandari, J.K. Chaar, M.J. Halliday, D.S. Moebus, B.K. Ray, and M.-Y. Wong. Orthogonal defect classification-a concept for in-process measurements. *Software Engineering, IEEE Transactions on*, 18(11):943 –956, nov 1992.
- [17] Jim Gray. Why do computers stop and what can be done about it?, 1985.
- [18] Michael Grottke and Kishor S. Trivedi. Fighting bugs: Remove, retry, replicate, and rejuvenate. *Computer*, 40(2):107 –109, feb. 2007.
- [19] Guojun Gan, Chaoqun Ma, and Wu Jianhong. *Data clustering: Theory, Algorithms, and Applications*. SIAM, Philadelphia, ASA, 2nd edition edition, 2007.
- [20] Kishor S. Trivedi. *Probability and statistics with reliability, queuing and computer science applications*. John Wiley and Sons Ltd., Chichester, UK, 2nd edition edition, 2002.
- [21] The JMP Statistical Tool SAS. <http://www.jmp.com>.
- [22] G. Carrozza, D. Cotroneo, R. Natella, A. Pecchia, and S. Russo. Memory leak analysis of mission-critical middleware. *J. Syst. Softw.*, 83:1556–1567, September 2010.
- [23] James Skarie, Biplob K. Debnath, David J. Lilja, and Mohamed F. Mokbel. Scrap: A statistical approach for creating a database query workload based on performance bottlenecks. In *Workload Characterization, 2007. IISWC 2007. IEEE 10th International Symposium on*, pages 183 –192, sept. 2007.

- [24] Mumtaz Ahmad, Ashraf Aboulnaga, and Shivnath Babu. Query interactions in database workloads. In *Proceedings of the Second International Workshop on Testing Database Systems*, DBTest '09, pages 11:1–11:6, New York, NY, USA, 2009. ACM.
- [25] Atzeni Paolo, ceri Paolo, Paraboschi Stefano, and Torlone Riccardo. Basi di dati, modelli e linguaggi di interrogazione. 2002.
- [26] The Jmeter Tool. <http://jakarta.apache.org/jmeter/>.
- [27] D. Cotroneo, S. Orlando, and S. Russo. Characterizing aging phenomena of the java virtual machine. In *Reliable Distributed Systems, 2007. SRDS 2007. 26th IEEE International Symposium on*, pages 127 –136, oct. 2007.

