UNIVERSITÀ DEGLI STUDI ROMA TRE

Dipartimento di Ingegneria Civile, Informatica
e delle Tecnologie Aeronautiche
Corso di Laurea in Ingegneria Informatica

Progetto di Tecnologie e Architetture per la Gestione dei Dati

Alessandro Schmitt 577421 Michele Guida 576985

Anno Accademico 2024/2025

Abstract

In questa relazione vengono modellate le prestazioni di un DBMS sulla base di una traccia di workload (`transaction.csv`), costituita da quattro attributi: *Transaction ID*, *CPU Time*, *I/O on Disk* e *% Read Operations*. La traccia, ottenuta osservando il DBMS per un intervallo di tempo $T = 24$ ore, presenta un'utilizzazione delle risorse pari al 23% per la CPU e al 19% per l'array RAID. Si realizza un modello analitico del sistema al fine di ricavare il tempo di risposta medio complessivo e per componente, nonché il numero medio di job presenti nel sistema. Successivamente, ipotizzando un incremento proporzionale su tutte le classi di workload, si determina il massimo carico per classe con cui sollecitare il sistema, mantenendo l'utilizzazione del bottleneck inferiore o uguale al 70%. A questo punto, si modifica la configurazione del sistema adottando un array di tipo RAID-1, e si risponde alla seguente domanda: "È possibile sostenere il throughput calcolato precedentemente con un'utilizzazione del collo di bottiglia $\leq 70\%$? Se sì, con quanti dischi? Se no, quali modifiche sono necessarie affinché il sistema soddisfi tali requisiti e abbia almeno un disco per la tolleranza ai guasti?". Nel seguito della relazione vengono illustrati tutti i passaggi svolti per giungere alle conclusioni.

Ambiente di esecuzione

Per la realizzazione di questo progetto è stato utilizzato principalmente *Java Modelling Tools* (JMT), una suite open source di strumenti grafici sviluppati in Java. JMT consente di modellare sistemi tramite reti di code (*Queueing Networks*), simulare o risolvere analiticamente modelli per stimare diverse metriche prestazionali, e visualizzare grafici e indicatori sulle performance del sistema.

Fase di Analisi dei Requisiti

È necessario caratterizzare il carico di lavoro al fine di affinare le successive valutazioni prestazionali. In questa fase, l'attenzione è rivolta all'individuazione delle diverse classi di workload. A partire dalla traccia di workload fornita, si procede inizialmente con un'analisi visiva, finalizzata all'individuazione di eventuali cluster e alla comprensione delle distribuzioni presenti nel dataset.

Statistiche

Tramite il tool *JWAT*, si esegue un'analisi statistica univariata per osservare alcune metriche significative.

Univariate statistics
This panel shows univariate statistics and graphs

Variable	Mean	Variance	Std. Dev.	Coeff. of var.	Minimum	Maximum	Range	Median	Kurtosis	Skewness	Num. Obs.
T-ID	500.005E2	833.342E6	288.677E2	577.347E-3	100.000E-2	100.000E3	999.990E2	500.010E2	-120.005...	631.862E...	100.000E3
CPU-Time	200.528E-2	811.580E-4	284.882E-3	142.066E-3	101.640E-2	313.200E-2	211.560E-2	196.450E-2	-351.223...	187.050E-3	100.000E3
I/O-Disk	300.037E-2	748.306E-4	273.552E-3	911.727E-4	200.000E-2	400.000E-2	200.000E-2	300.000E-2	103.632E-1	140.179E-4	100.000E3
Ratio-Read	500.026E-1	161.728E1	402.154E-1	804.266E-3	000.000E0	100.000E0	100.000E0	500.000E-2	-194.208...	-115.285...	100.000E3

Figure 1: Analisi Univariata tramite JWAT

A questo punto tramite gli scatterplot, realizzati in Python, si verifica visivamente la presenza di cluster, approccio semplice ed efficiente che merita poi di essere esteso attraverso un indagine più approfondita:

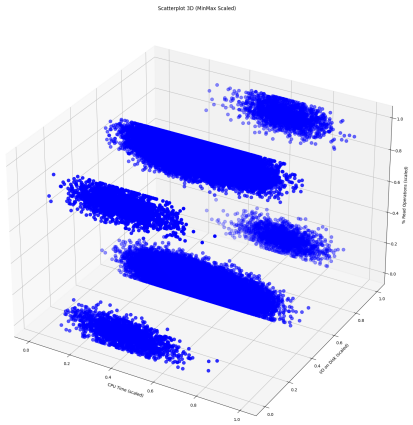

Figure 2: Scatter Plot per individuare una possibile clusterizzazione del dataset, utilizzando tutte le features escluso il Transaction ID

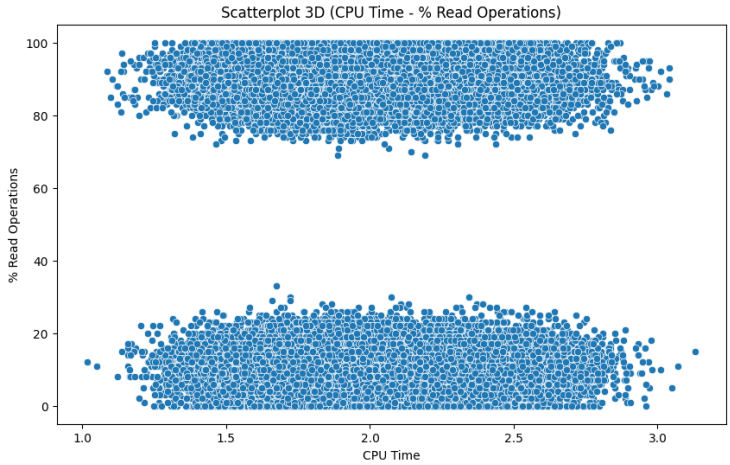

Figure 3: Ulteriore Scatter Plot per verificare possibili alternative

Questo approccio viene successivamente confrontato con i risultati ottenuti tramite l'algoritmo K-Means, tramite il tools JWAT (Fig. 4), che consente di identificare automaticamente gruppi di transazioni con caratteristiche simili.

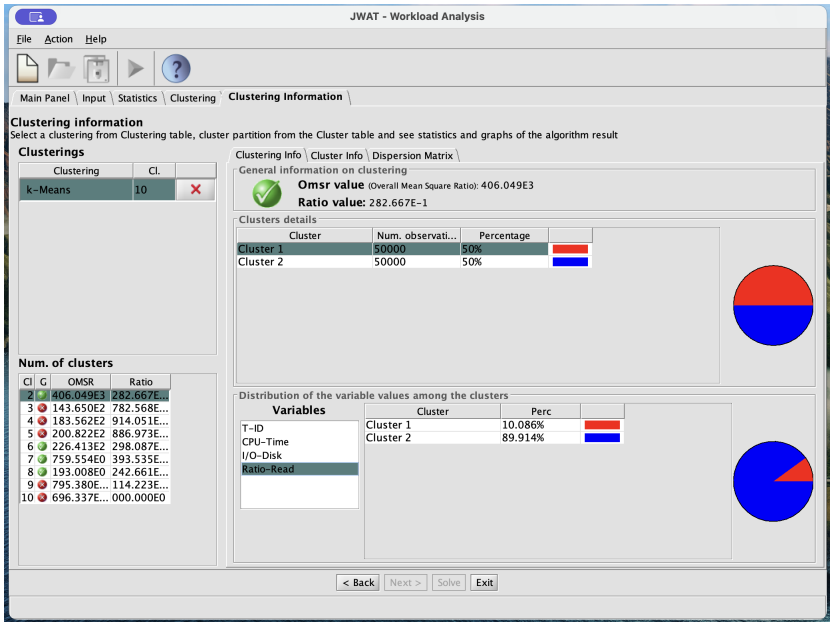

Figure 4: Clustering info del Cluster numero 1 rispetto alla feature discriminante '%Read Operations'

Come è possibile notare dall'analisi visiva, il numero ottimale di cluster può essere identificato in corrispondenza di k=2 o k=6, valori per i quali l'algoritmo k-means raggiunge le performance migliori. Tra queste opzioni, si è scelto di adottare **k=2** poiché questa configurazione genera due sottogruppi di transazioni internamente omogenei che si differenziano in modo significativo rispetto alla percentuale di operazioni di lettura. La scelta di scartare k=6, pur rappresentando un'alternativa tecnicamente valida, è motivata da una segmentazione eccessiva che produce 4 cluster su 6 scarsamente rappresentativi. L'implementazione di un numero superiore di cluster comporterebbe un incremento della complessità analitica senza fornire vantaggi concreti per la modellazione. È importante sottolineare che nei primi due task, le due classi si comportano in maniera identica. Ciò avviene perché, con la configurazione iniziale basata su RAID-0, sia le operazioni di lettura che quelle di scrittura beneficiano degli stessi vantaggi in termini di parallelizzazione, rendendo la percentuale di operazioni di lettura un parametro ininfluente sulle prestazioni complessive del sistema. Solo successivamente, con la sostituzione dell'array RAID-0 con un RAID-1, emergono differenze significative tra le due classi. In questa configurazione, le operazioni di lettura continuano a beneficiare della parallelizzazione, mentre le operazioni di scrittura risultano penalizzate dalla necessità di replicazione dei dati su entrambi i dischi, rendendo così significativa la diversa composizione percentuale delle operazioni di lettura tra i cluster solo nell'ultimo task. Alla luce di queste considerazioni, pur osservando comportamenti inizialmente equivalenti tra le due classi, si è ritenuto opportuno preservare la distinzione in classi di workload per l'intera durata della sperimentazione, al fine di mantenere un framework analitico coerente e poter catturare le divergenze prestazionali attese nella configurazione finale.

Informazioni per ciascun cluster:

Cluster Information
This panel shows information of variables (center and statistics) within a single cluster

Cluster 1
Cluster 2

Cluster 1/2 has 50000 observations

Sel.	Name	Center	Std. Dev.	Kurt.	Skew.
☐	T-ID	502.011E2	288.395E2	-119.857E-2	-959.086E-5
☑	CPU-Time	200.734E-2	286.557E-3	-349.691E-3	192.207E-3
☑	I/O-Disk	300.118E-2	273.386E-3	103.788E-1	448.014E-4
☑	Ratio-Read	100.867E-1	490.347E-2	-255.217E-3	119.218E-3

Figure 5: Cluster info del Cluster numero 1

Cluster Information
This panel shows information of variables (center and statistics) within a single cluster

Cluster 1
Cluster 2

Cluster 2/2 has 50000 observations

Sel.	Name	Center	Std. Dev.	Kurt.	Skew.
☐	T-ID	497.999E2	288.947E2	-120.127E-2	961.549E-5
☑	CPU-Time	200.322E-2	283.186E-3	-354.241E-3	181.178E-3
☑	I/O-Disk	299.956E-2	273.718E-3	103.470E-1	-166.334E-4
☑	Ratio-Read	899.185E-1	489.119E-2	-222.858E-3	-125.223E-3

Figure 6: Cluster info del Cluster numero 2

Le informazioni rilevate per ciascun cluster consentono di definire le corrispondenti classi di workload, utili alla costruzione di un modello di prestazione con il seguente livello di astrazione: Analitico, Multi-Classe, Classi Aperte.

Parametrizzazione

Classes

Name	Type	Population	Arrival Rate
Class1	open		0.578703704
Class2	open		0.578703704

Stations

Name	Type
CPU	Load Independent
Disk1	Load Independent
Disk2	Load Independent

Figure 7: Modello di Prestazione analitico, con 2 classi aperte con Arrival Rate pari a 0.5786..., 3 componenti principali: CPU, Disk1, Disk2 (per modellare l'Array RAID-0)

Per una parametrizzazione del modello, è necessario calcolare le metriche di performance.

- **Throughput del sistema** (X_0):

$$X_0 = \frac{\text{numero di transazioni}}{T} = \frac{100.000}{86400s} = 1.1574 \ tps$$

  dove T è l'intervallo di osservazione (nel nostro caso, 24 ore).

- **Throughput per classe** $X_{0,r}$:

$$X_{0,r} = X_0 \cdot \frac{n_r}{n}$$

  - n_r è il numero di campioni appartenenti alla classe r
  - n è il numero totale di campioni

  allora:

$$X_{0,1} = X_{0,2} = 1,1574 \cdot \frac{50.000}{100.000} = 0.5787 \ tps$$

- **Utilizzazione per classe** $U_{i,r}$:

$$U_{i,r} = U_i \cdot f_{i,r}$$

Poiché le classi sono equinumerose e i valori assunti sono i medesimi, ognuna contribuisce con un peso pari a 0.5, rendendo superflua l'esplicitazione tramite formule.

$$f_{\text{CPU},0} = f_{\text{CPU},1} = f_{\text{RAID},0} = f_{\text{RAID},1} = 0,5$$

Quindi le utilizzazioni sono:

$$U_{\text{CPU},0} = U_{\text{CPU},1} = 11.5\%$$

$$U_{\text{RAID},0} = U_{\text{RAID},1} = 9.5\%$$

- **Service Demand** $D_{i,r}$:

$$D_{i,r} = \frac{U_{i,r}}{X_{0,r}}$$

Quindi i Service Demand sono:

$$D_{\text{CPU},0} = D_{\text{CPU},1} = 0.198s$$

$$D_{\text{RAID},0} = D_{\text{RAID},1} = 0.164s$$

1 Soluzione Task 1

Utilizzando lo strumento JMVA opportunamente configurato, è possibile rispondere alle seguenti domande:

(a) Tempo medio di risposta:

- Sistema: $0.4395s$
- CPU: $0.2581s$
- DISK 0 = DISK 1 : $0.0907s$

(b) Il numero medio di job presenti nel sistema è: 0.5086

Task 2 - Incremento Proporzionale del Workload

La metodologia prosegue con l'individuazione del bottleneck; nel presente caso di studio risulta essere la CPU. L'identificazione è stata effettuata tramite l'analisi dell'utilizzo, benché si sarebbe potuta ottenere equivalentemente attraverso la valutazione del Service Demand. Attraverso l'analisi What-If è stato possibile osservare la variazione dell'utilizzo della CPU (risorsa bottleneck) in risposta a un incremento proporzionale applicato a tutte le classi di workload.

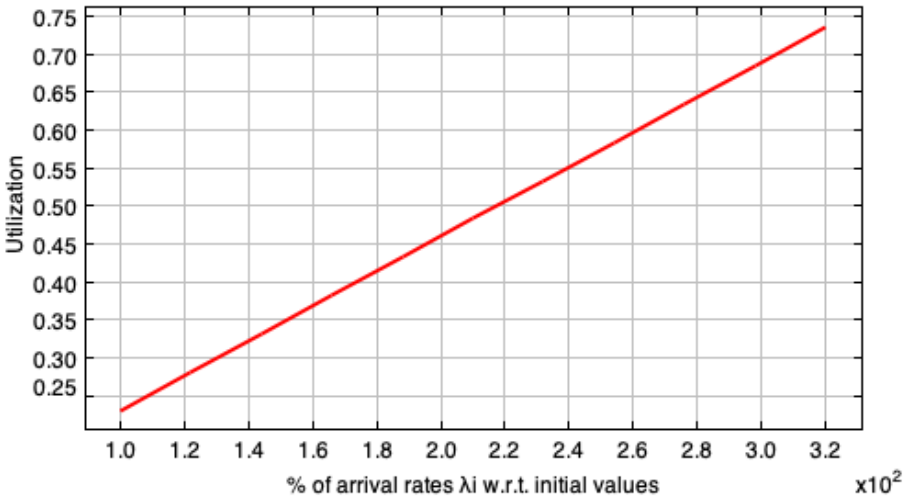

Figure 8: Andamento dell'utilizzazione in vista di un incremento proporzionale su tutte le classi del workload

2 Soluzione Task 2

Attraverso l'analisi grafica, fissando una soglia di utilizzazione del 70%, è stata determinata l'intersezione con la retta. Il punto identificato indica un valore sull'asse delle ascisse corrispondente al 304.35% del tasso d'arrivo iniziale 0.5787 *tps*, equivalente ad un **carico massimo di 1.7612 *tps* per classe** di workload.

Task 3 – Sostituzione RAID-0 con RAID-1

Nella fase conclusiva del workflow, si procede alla sostituzione dell'array `RAID-0` con una configurazione `RAID-1`. L'obiettivo è determinare se il sistema sia in grado di sostenere il throughput precedentemente calcolato, rispettando il vincolo di utilizzazione del collo di bottiglia non superiore al 70%. La configurazione `RAID-1` presenta caratteristiche prestazionali distinte rispetto al `RAID-0`, particolarmente evidenti nella gestione differenziata delle operazioni di lettura e scrittura:

- **Operazioni di lettura**: beneficiano della parallelizzazione tra i dischi dell'array, con conseguente riduzione del Service Demand proporzionale al numero di dischi N:

$$D^{(\text{RAID-1})}_{\text{DiskX, lettura}} = \frac{D_{\text{RAID-0}}}{N} \cdot \%\text{Read Operations}$$

  dove $D^{(\text{RAID-1})}_{\text{DiskX, lettura}}$ rappresenta il Service Demand per il disco X dell'array RAID-1 relativo alle operazioni di lettura. Tale comportamento risulta analogo a quello osservato nella configurazione RAID-0.

- **Operazioni di scrittura**: richiedono la replicazione dei dati su tutti i dischi dell'array, eliminando i vantaggi della parallelizzazione:

$$D^{(\text{RAID-1})}_{\text{DiskX, scrittura}} = D_{\text{RAID-0}} \cdot \%\text{Write Operations}$$

  dove $D^{(\text{RAID-1})}_{\text{DiskX, scrittura}}$ indica il Service Demand per il disco X dell'array RAID-1 relativo alle operazioni di scrittura. Questa configurazione comporta una penalizzazione prestazionale rispetto al RAID-0, dove la parallelizzazione è applicabile anche alle operazioni di scrittura.

Considerando i parametri operativi di throughput per la classe $X_{0,r} = 1.7612$ (identico per entrambe le classi di workload) e il Service Demand di riferimento $D_{\text{RAID-0}} = 0.16420$, si ottengono i seguenti valori:

$$D^{(\text{RAID-1})}_{\text{Disk0,0}} = D^{(\text{RAID-1})}_{\text{Disk1,0}} = 0.1559s$$
$$D^{(\text{RAID-1})}_{\text{Disk0,1}} = D^{(\text{RAID-1})}_{\text{Disk1,1}} = 0.0903s$$

L'analisi viene condotta nel rispetto del vincolo prestazionale:

$$U_{\text{Bottleneck}} \leq 0.7$$

si ottiene:

Utilization
Utilization of a customer class at the selected statior

*	*Aggregate*	Class1	Class2
Aggregate	--	--	--
CPU	0.7000	0.3503	0.3496
Disk1	0.2891	0.1446	0.1445
Disk2	0.2891	0.1446	0.1445

Figure 9: Utilizzazione con Array RAID-0

Utilization
Utilization of a customer class at the selected stati
number of customers in the station (it may be grea

*	*Aggregate*	Class1	Class2
Aggregate	--	--	--
CPU	0.7000	0.3503	0.3496
Disk1	0.4337	0.2746	0.1591
Disk2	0.4337	0.2746	0.1591

Figure 10: Utilizzazione con Array RAID-1

3 Soluzione Task 3

La transizione da RAID-0 a RAID-1 modifica sostanzialmente l'utilizzo dei dischi, ma lascia inalterata l'utilizzazione della CPU, che rimane ancora il collo di bottiglia del sistema. Di conseguenza, nonostante la modifica nella configurazione, il throughput precedentemente stimato resta sostenibile mantenendo l'utilizzazione del bottleneck $\leq 70\%$. Pertanto questo vincolo operativo può essere soddisfatto impiegando solo **2 dischi**, come nella configurazione RAID-0 originale.