[1] Confronto Sperimentale tra MongoDB e CouchDB tramite analisi di recensioni Amazon

COURSE NAME

Big Data Analytics

Author:

Michellini Marco (207429)

June 6, 2025

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

[1] https://github.com/gammic/BD_Progetto.git

Contents

Abstract

In questa relazione viene presentato un confronto tra due sistemi di gestione di database NoSQL: MongoDB e CouchDB. Entrambi seguono un modello document-based, ma si differenziano per architettura, modalità di interrogazione e gestione dei dati. La sperimentazione pratica consiste nell'eseguire un insieme di query analitiche su un dataset contenente recensioni di prodotti, valutando le prestazioni di ciascun sistema. I risultati ottenuti sono stati confrontati in termini di tempo di risposta, evidenziando i punti di forza e le limitazioni di ogni sistema.

1 Introduzione ai Sistemi NoSQL: MongoDB e CouchDB

Nel contesto dell'elaborazione di grandi volumi di dati, i database NoSQL offrono una valida alternativa ai sistemi relazionali tradizionali. In questa relazione si sono analizzati e confrontati due sistemi rappresentativi di questo paradigma: MongoDB e CouchDB.

MongoDB è un database *document-based* che utilizza una struttura di dati in formato BSON (una estensione binaria di JSON). È pensato per offrire scalabilità orizzontale, flessibilità nello schema dei dati e alte prestazioni nelle operazioni di lettura e scrittura. MongoDB supporta nativamente le query complesse, gli indici su qualsiasi campo, e aggregazioni complesse tramite il framework Aggregation Pipeline.

CouchDB, anch'esso *document-based*, adotta il formato JSON per la memorizzazione dei dati, con una forte enfasi sull'affidabilità e sulla sincronizzazione tra nodi. CouchDB implementa *map-reduce* per l'esecuzione di query e viste definite dall'utente e segue un'architettura RESTful basata su HTTP. Il suo approccio alla gestione della concorrenza e alle revisioni dei documenti lo rende adatto per applicazioni distribuite con requisiti di sincronizzazione offline.

Le differenze tra i due sistemi emergono soprattutto nel modello di query: MongoDB privilegia flessibilità e potenza espressiva lato client, mentre CouchDB promuove un'esecuzione distribuita e la persistenza delle viste, con aggiornamenti incrementali.

2 Metodologia

Il confronto tra MongoDB e CouchDB è stato condotto utilizzando un dataset di recensioni di prodotti, ciascuna caratterizzata da attributi come identificativo del prodotto, identificativo utente, punteggio assegnato, data, sommario e testo della recensione.

Il dataset è stato caricato in entrambi i sistemi mantenendo la stessa struttura dei documenti JSON. Sono state formulate otto query analitiche con l'obiettivo di:

- Calcolare il numero totale di recensioni
- Individuare i prodotti con più recensioni
- Trovare gli utenti più attivi
- Analizzare la distribuzione delle valutazioni
- Identificare recensioni con alto rapporto di utilità
- Calcolare il punteggio medio per prodotto
- Analizzare il trend temporale delle recensioni
- Valutare la lunghezza media delle recensioni per punteggio

Nel caso di MongoDB, le query sono state espresse attraverso l'Aggregation Framework. In CouchDB, invece, sono state create delle *map-reduce views* all'interno di *design document*, una per ogni tipo di analisi, e interrogate tramite richieste HTTP REST.

Per ogni query, sono state effettuate tre misurazioni dei tempi di esecuzione in entrambi i sistemi per calcolare una media temporale e ottenere una valutazione più stabile delle performance.

Nella prossima sezione verranno riportati i risultati ottenuti e confrontati tra i due sistemi.

3 Risultati

Nel corso della sperimentazione sono state eseguite otto query su entrambi i sistemi (CouchDB e MongoDB) per valutare le prestazioni su un dataset costituito da recensioni di prodotti. Ogni query è stata eseguita tre volte ed è stata calcolata la media dei tempi di risposta. I risultati sono riportati di seguito e da ciascuno di essi vengono dedotte le differenze tra i due sistemi:

- **Totale recensioni**:
  - CouchDB: 0.01 s
  - MongoDB: 0.14 s

In questa prima query viene conteggiato il numero totale di recensioni. Couch risulta molto rapido, in quanto viene eseguito tramite una chiamata unica a *doc_count*, mentre Mongo risulta leggermente più lento, dato che deve eseguire una count che legge sul *query engine*. In questo caso la chiave vincente è la leggerezza e la struttura dei documenti su Couch.

- **Prodotti più recensiti**:
  - CouchDB: 3.11 s
  - MongoDB: 0.28 s

  Questa query ricerca i prodotti più recensiti tramite raggruppamento delle recensioni per *productId* e conteggio per ognuno della quantità di recensioni. MongoDB sfrutta come punto di forza il framework di aggregazione nativo che supporta operatori come *$group* e *$sort*, mentre su Couch questo non è presente, ed è necessario creare una view *map-reduce* che sfrutta l'attributo *_sum* per conteggiare la quantità di recensioni per prodotto.

- **Utenti con più recensioni**:
  - CouchDB: 5.91 s
  - MongoDB: 0.63 s

  Tramite questa query vengono ricercati gli utenti che hanno effettuato più recensioni, e si presenta lo stesso caso della query precedente, ovvero ottimizzazione nativa per operazioni di raggruppamento per Mongo, mentre necessità di funzioni *map-reduce* per Couch.

- **Distribuzione valutazioni**:
  - CouchDB: 0.01 s
  - MongoDB: 0.19 s

  In questo caso, dove vengono restituite per ciascuna valutazione (1-5 stelle) la relativa quantità di recensioni, Couch risulta molto rapido perchè, nonostante la struttura della view sia uguale a quelle precedenti, in questo caso il numero distinto delle chiavi emesse è pari a 5, e Couch sfrutta cache interne che mantengono salvata la view per avere un tempo di risposta minimo. Nei casi precedenti invece si ottengono moltissime chiavi e quindi la durata della query aumenta esponenzialmente.

- **Recensioni utili**:
  - CouchDB: 8.58 s
  - MongoDB: 0.75 s

  Qui l'obiettivo è ottenere l'utilità della review, calcolandola come valutazioni positive su totale valutazioni. Questo porta ad accentuare ancora maggiormente il problema di Couch nel processare molte chiavi, che qui è la coppia prodotto-utente. Inoltre non supporta calcoli dinamici nella view, quindi il calcolo di questo rapporto viene fatto lato client. Invece Mongo svolge tutto con tempi leggermente maggiori delle query precedente, dovuto all'uso di un'unica query in *pipeline* che raggruppa, calcola il rapporto e ordina i risultati.

- **Prodotti migliori (media valutazioni)**:
  - CouchDB: 2.80 s
  - MongoDB: 0.53 s

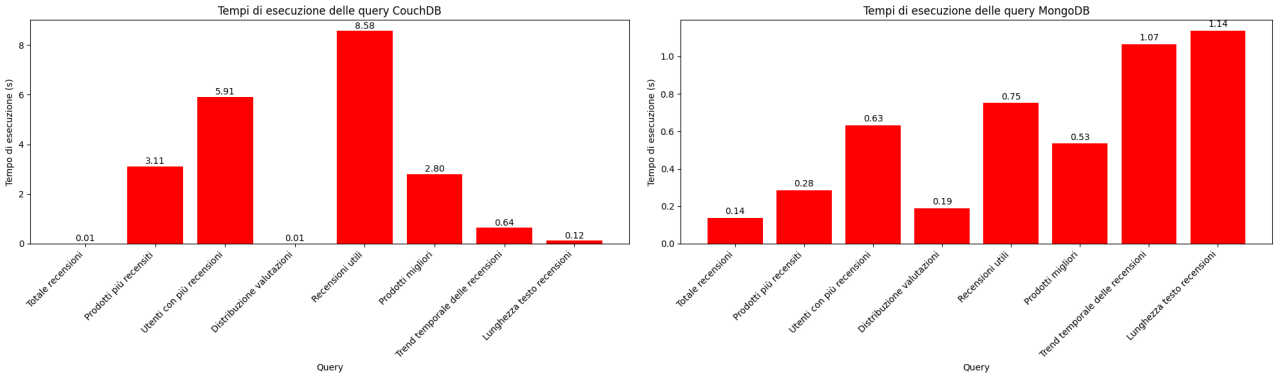
  La query in questione ritorna i 10 prodotti con miglior valutazione media (solamente prodotti con almeno 50 recensioni). Nuovamente si ricade nel problema di Couch che, dovendo processare molte chiavi, ha un tempo di risposta più alto. Inoltre come citato precedentemente non supportando calcoli nella view, essa ritorna tramite il parametro *_stats* i valori di somma e conteggio delle valutazioni, permettendo il calcolo della media lato client. Invece Mongo permette tutto ciò nella singola query, anche il vincolo dei prodotti con almeno 50 recensioni.

- **Trend temporale delle recensioni**:
  - CouchDB: 0.64 s
  - MongoDB: 1.07 s

In questa query, che restituisce il trend temporale delle recensioni (grafico 2), risulta migliore Couch. Questo è dovuto dalla minore quantità di chiavi da processare ma soprattutto dall'automatica organizzazione dei campi temporali, che vengono mantenuti ordinati da Couch, e quindi la successiva reduce è molto ottimizzata. Su Mongo invece non si indicizza direttamente l'aggregazione per data, e in assenza di questo indice il sistema deve fare una full scan e quindi spendere più tempo.

- **Lunghezza testo recensioni per valutazione**:
  - CouchDB: 0.12 s
  - MongoDB: 1.14 s

  L'ultima query è quella che premia maggiormente Couch rispetto a Mongo. Calcolando direttamente la lunghezza del testo in fase di indicizzazione infatti, il valore è poi già pronto per la reduce. Inoltre è stata implementata una reduce personalizzata che aggrega efficientemente *total* e *count*, in modo da calcolare facilmente la media lato client con una semplice divisione. Un altro punto a favore è di nuovo la bassa cardinalità di *Score* (1-5 stelle), e quindi la struttura della view è compatta e snella da utilizzare. Infine sfruttando nuovamente il caching di Couch la risposta è quasi istantanea.



(a) Tempi di esecuzione di CouchDB

(b) Tempi di esecuzione di MongoDB

Figure 1: Tempi di esecuzione

Come si può osservare anche dai grafici 1, riassumendo, MongoDB ha generalmente tempi di risposta inferiori rispetto a CouchDB, con l'eccezione delle query sul trend temporale e sulla lunghezza delle recensioni, dove CouchDB si è dimostrato più efficiente.

4 Conclusioni

L'analisi comparativa tra MongoDB e CouchDB ha permesso di valutare le prestazioni e le funzionalità offerte dai due database NoSQL all'interno di un contesto di elaborazione e analisi di recensioni testuali.

4.1 Prestazioni

1. **MongoDB** ha mostrato tempi di esecuzioni inferiori nella maggior parte delle query, in particolare in quelle che richiedono ordinamenti complessi, raggruppamenti su attributi con altà cardinalità (come *product* o *user*), o calcoli dinamici come medie o filtri personalizzati;
2. **CouchDB** è risultato superiore in alcune query, quando le view erano già calcolate e ben progettate, nel caso quindi delle ultime due. Ciò dimostra che, in caso di pre-aggregazione opportuna tramite *map-reduce*, CouchDB può fornire risposte molto rapide.

4.2 Analisi qualitativa

Il principale *bottleneck* di Couch risiede nell'alta cardinalità delle chiavi emesse, producendo di conseguenza tempi per scansionare più lunghi. Al contrario query aggregate su campi e con bassa cardinalità (come *Score*) sono molto efficienti.

Mongo invece, basandosi su aggregation pipelines e ottimizzazioni a livello di *query engine*, gestisce meglio operazioni ad hoc e trasformazioni complesse. Tuttavia, l'assenza di caching e pre-aggregazione implica un maggior carico computazionale ad ogni richiesta, specialmente senza indici specifici.

4.3 Considerazioni finali

CouchDB è più adatto a scenari con query ripetitive su dati statici o poco variabili, dove si possono progettare view efficienti anticipatamente. **MongoDB** si dimostra più versatile e performante in scenari dinamici e con analisi esplorative complesse, a scapito però di un maggior sfruttamento di risorse a runtime. Pertanto la scelta tra i due dipende fortemente dal tipo di applicazione: per dashboard statiche o sistemi embedded CouchDB è competitivo; per analisi dinamiche o interrogazioni flessibili MongoDB rappresenta la scelta preferibile.

References

[1] Kristina Chodorow. *MongoDB: The Definitive Guide*, 2nd Edition. O'Reilly Media, 2013. Disponibile su: `https://www.mongodb.com/docs/manual/`

[2] Apache Software Foundation. *Apache CouchDB Documentation*. 2024. Disponibile su: `https://docs.couchdb.org/`

[3] Michael Stonebraker. "SQL databases v. NoSQL databases." *Communications of the ACM*, vol. 53, n. 4, pp. 10–11, 2010.

A Additional graphs and data

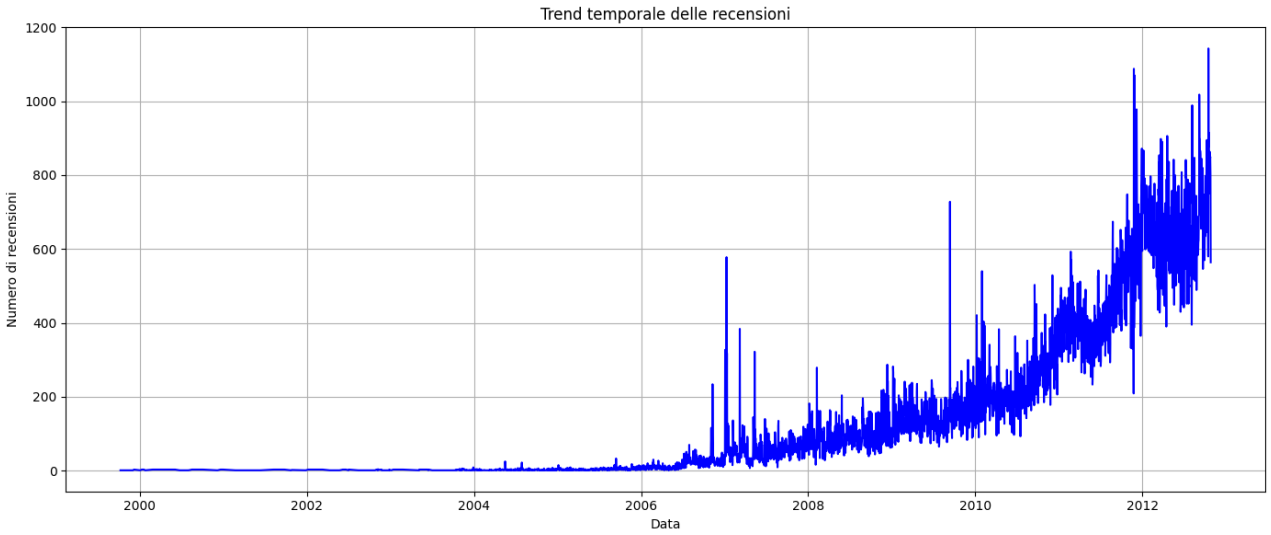


Figure 2: Trend temporale delle recensioni