Architettura di MongoDB e proprietà ACID

Dr Sara Migliorini

Architettura di MongoDB

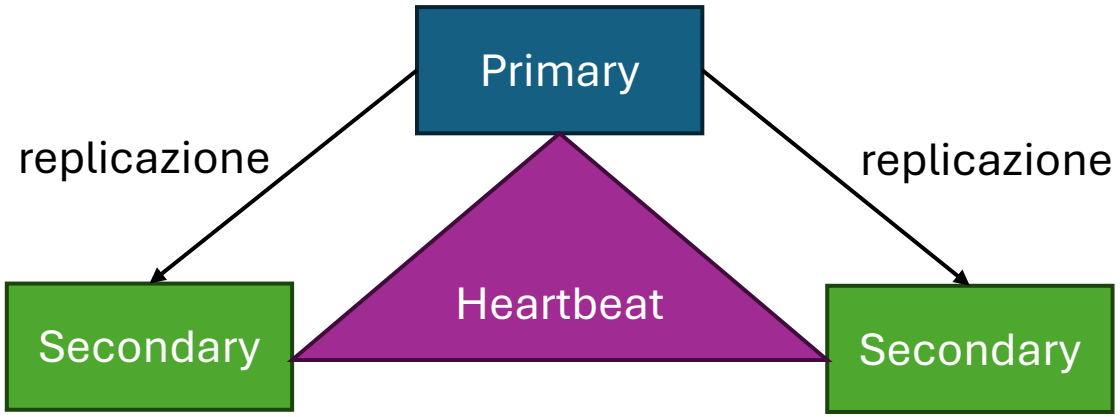
- L’architettura di MongoDB si basa su due concetti fondamentali: replicazione e sharding (frammentazione).
- Replicazione:
  - Componente fondamentale dell’architettura distribuita di MongoDB
  - Garantisce accessibilità ai dati e resilienza in caso di guasti
  - Consiste nel disseminare copie identiche degli stessi dati tra server diversi, salvaguardando la disponibilità dell’informazione in caso di fallimento o guasto di un singolo server (ridondanza e affidabilità).

Architettura di MongoDB

- Sharding:
  - Consiste nel suddividere un dataset di grandi dimensioni in parti più piccole e più maneggevoli.
  - Ciascun shard memorizza una porzione del dataset complessivo in una diversa istanza del database server (macchina fisica diversa).
  - Strategia di scalabilità orizzontale per aumentare le performance.
  - Ogni shard può anche essere replicato per garantire l'integrità e la disponibilità.

Replicazione: Replica set

- In MongoDB un replica set è una collezione di processi mongod che mantengono una copia di uno stesso dataset.
  - Ridondanza e alta disponibilità
  - Tolleranza ai guasti rispetto al failure di un server
  - Possibilità di aggiornare il software o hardware di un nodo, senza interruzioni di servizio
  - Migliora le prestazioni delle operazioni di scrittura



Replicazione: Replica set

- Il nodo primario gestisce tutte le operazioni di scrittura e mantiene un log dei cambiamenti intervenuti nel dataset (operation log: oplog)
- Ciascun replica set può avere un solo nodo primario
- I nodi secondari replicano le operazioni scritte nell'oplog del nodo primario all'interno delle proprie copie del dataset
- Nel caso il nodo primario diventi non disponibile, uno nodo secondario qualificato viene eletto nodo primario.

Replicazione: Algoritmo di elezione

- L’elezione di un noto all’interno di un replica set avviene tramite un protocollo basato sull’algoritmo di consenso Raft.
- Questo protocollo prevede un meccanismo di voto che permette di selezionare quale membro del replica set assumerà il ruolo di nodo primario.
- L’algoritmo di elezione viene attivato nei seguenti casi:
  - Aggiunta o rimozione di un nodo dal replica set
  - Inizializzazione di un replica set
  - Un errore di comunicazione tra qualsiasi dei nodi secondari ed il nodo primario che eccede il timeout prestabilito (default 10 sec).
    - Identificazione automatica della perdita del nodo primario

Replicazione: Algoritmo di elezione

- Quando il nodo primario diventa non più disponibile, i nodi secondari eseguono una votazione per determinare un nuovo nodo primario.
  - La votazione tipicamente non richiede più di 12 secondi (inclusa la fase in cui viene stabilita la non raggiungibilità del nodo primario, fino al termine dell'elezione).
- Il nodo replica con il timestamp di scrittura più recente è quello con maggiori probabilità di vincere l'elezione.
  - Minimizza la probabilità di dover eseguire un rollback nel caso il nodo primario precedente ritorni disponibile.
- Dopo aver concluso un'elezione, i nodi entrano di un periodo di congelamento, durante il quale non possono iniziare un'altra elezione.
  - Permette di evitare elezioni continue e rapide, che possono destabilizzare l'intero sistema.
- Il replica set non può eseguire operazioni di scrittura, fintantoché l'elezione del nuovo nodo primario non si è conclusa con successo.

Replicazione: oplog

- L'oplog è un file contenente il log di tutte le operazioni effettuate sui database.
- Le operazioni di scrittura sono prima eseguite sul nodo primario e trascritte sul suo oplog. Quindi i nodi secondari replicano le stesse operazioni di scrittura in modo asincrono.
  - Ogni membro della replica mantiene un proprio file di oplog, in modo da poter determinare la relazione tra lo stato del nodo e lo stato complessivo del database.
- Per supportare la replicazione, è necessario che tutti i membri del replica set si scambino informazioni attraverso la condivisione dei propri oplog.
- Le operazioni contenute nell'oplog sono idempotenti.

Replicazione: Architettura di un replica set

- La configurazione tipica prevede la presenza di 3 membri all'interno di ciascun replica set.
- Best practice:
  - Assicurarsi che i replica set siano composti da un numero dispari di membri votati, in modo da evitare situazioni di stallo dovute a parità di voti.
  - In caso di un numero pari di membri votanti, prevedere la presenza di un arbitro.
  - Includere la presenza di membri nascosti o ritardati, al fine di assegnare loro particolari task, come backup e reporting.
  - Assicurarsi che almeno un membro di ciascun replica set si trovi in un data center alternativo.

Replicazione: Arbitro

- L’arbitro è particolarmente utile nel caso in cui per questioni di risorse disponibili ci siano solo un nodo primario ed un nodo secondario.
- L’arbitro interviene durante l’elezione del nodo primario, ma non mantiene una copia del dataset e non può diventare un nodo primario.

Replicazione: Nodi nascosti

- I nodi nascosti non possono diventare nodi primari e non sono direttamente accessibili dalle applicazioni client.
- Possono partecipare e votare durante le elezioni.
- Vengono tipicamente usati per gestire task specifici, come i backup e il reporting.
- I nodi nascosti possono anche essere ritardati, cioè essere predisposti per replicare le operazioni del nodo principale con un certo ritardo pre-impostato.
    - Rappresentano una fonte di backup continuo e mantengono uno stato storico del dataset online.

Replicazione: Scritture

- Per garantire l'affidabilità delle repliche, è necessario stabilire un meccanismo in grado di assicurare che i dati siano segnalati come scritti solo quando un certo numero di nodi ha effettuato tale scrittura.
- Write concern:
  - w: numero di nodi del replica set che devono aver confermato la scrittura prima che tale operazione sia considerata conclusa
    - 0, 1 (solo nodo principale), majority, <number>
    - Un valore basso di w riduce la latenza, ma diminuisce anche il livello di sicurezza, un valore alto (majority) garantisce la persistenza del dato.
  - j: richiede conferma che l'operazione di scrittura sia stata scritta su disco.
  - wtimeout: limite di tempo (millisecondi) per l'operazione di scrittura oltre al quale viene ritornato un errore, permette di evitare che una scrittura blocchi il funzionamento per troppo tempo.

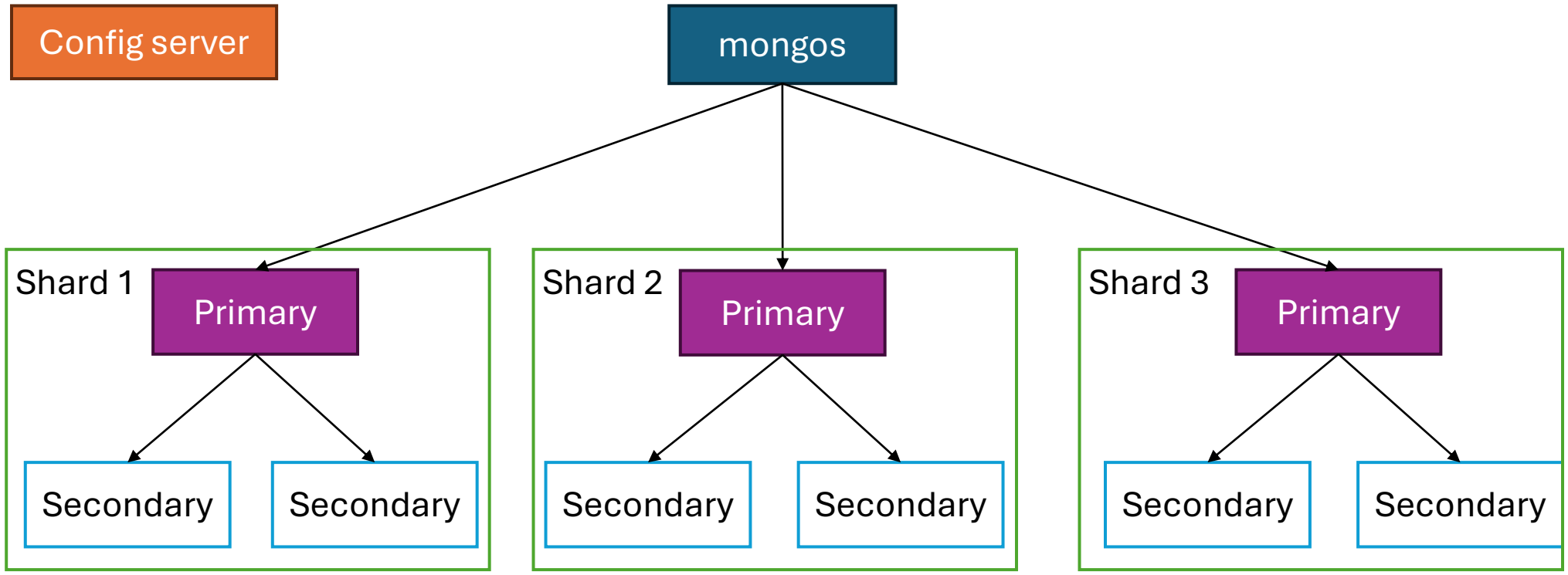
Replicazione: letture

- Di default, Mongo DB redirige le operazioni di lettura sul nodo primario.
- È possibile specificare che le letture vengano distribuite su più nodi per bilanciare il carico ed aumentare le prestazioni.
  - primary, primaryPreferred, secondary, secondaryPreferred, nearest
- È necessario garantire la consistenza e l'isolamento dei dati letti da una copia (read concern):
  - local: ritorna il dato disponibile più recente nel nodo al momento in cui viene eseguita la query, indipendentemente dallo stato di replicazione
  - available: ritorna il dato correntemente disponibile (latenza minima)
  - majority: ritorna il dato che è stato confermato dalla maggioranza dei nodi nel replica set (alta consistenza)
  - linearizable: ritorna il dato che è stato confermato da tutti i nodi del replica set (consistenza massima)
  - snapshot: ritorna i dati confermati dalla maggioranza dei nodi di un certo istante temporale (isolamento in caso di scritture concorrenti, visione coerente durante le transazioni)

Sharding

- Consiste nel dividere un database grande in parti o componenti più piccole che vengono disseminate in varie istanze del server.
  - shard: un replica set che memorizza una porzione dei dati di un cluster.
  - mongos: funge da distributore delle query, reindirizzando le richieste in modo efficiente tra i vari shard, consolida i risultati parziali ottenuti dagli shard producendo la risposta definitiva
  - config server: mantiene i metadati relativi allo sharding
- Il meccanismo di sharding offre un meccanismo di scalabilità orizzontale.
  - Scalabilità verticale: incrementare la potenza di calcolo di un singolo server.
  - Scalabilità orizzontale: suddivisione del carico di lavoro tra diversi server.

Sharding


Config server
mongos
Shard 1
Primary
Secondary
Secondary
Shard 2
Primary
Secondary
Secondary
Shard 3
Primary
Secondary
Secondary

Sharding

- Lo scharding viene implementato a livello di collezione.
- In talune situazioni è possibile implementare un micro-sharding, in cui più shard sono posizionati all'interno di uno stesso host.
- Vantaggi:
  - Aumentare la velocità di lettura e scrittura: esecuzione parallela
  - Aumentare la capacità di storage complessiva
  - Sfruttare la località del dato (zone sharding)

Sharding: shard key

- La distribuzione del dato avviene attraverso l'utilizzo di una shard key ottenuta a partire da una o più proprietà dei documenti.
- I documenti vengono suddivisi in chunk non sovrapposti sulla base del valore della chiave di shard.
  - A partire da MongoDB 4.4 è gestita la possiblità di avere valori di chiave nulli (o alcuni componenti nulli)
  - A partire da MongDB 4.4 è possibile aumentare la chiave di shard includendo anche dei prefissi o suffissi al valore delle proprietà
  - A partire da MongoDB 5.0 è possibile eseguire il reshard di una collezione, cambiando la chiave

Sharding: strategie

- Sono possibili due strategie per implementare la suddivisione dei documenti in chunk:
- Ranged sharding: i dati vengono suddivisi in base ad intervalli continui di valori della chiave di shard
  - Ottimale per implementare range query sui valori della chiave
  - Metodo di default
  - La cardinalità del dominio della shard key permette di determinare il numero massimo di chunck che possono essere prodotti
  - Se ci sono alcuni valori più probabili di altri è possibile ottenere dei chunk più numerosi che rappresentano dei colli di bottiglia
  - Una chiave di shard basata su valori che crescono o decrescono in modo monotonico, producono più probabilmente delle operazioni di inserimento che coinvolgono un unico chunk.

Sharding: strategie

- Hashed sharding: opera calcolando il valore di hash della chiave di shard e assegnando ciascun chunk in base ad un range determinato da questo valore di hash.
  - Valori di chiave simile, avranno un valore di hash molto diverso (diverso chunk).
  - Non è efficiente in caso di operazioni di tipo range-query

Sharding: chunks e bilanciatore

- Il balancer è un processo che lavora in background al fine di assicurare che la quantità di dati assegnati a ciascun shard sia la più bilanciata possibile.
- Quando la quantità di dati assegnata ad un particolare shard raggiunge determinati livelli, il balancer tenta di redistribuire tali dati tra altri shard, al fine di promuovere un'equa redistribuzione dei dati, mantenendo il principio di località
- Jumbo chunk: chunk che eccede la dimensione massima ammessa, ma non può essere suddiviso in automatico, richiede un intervento manuale
- Chunk indivisibile: chunk caratterizzato da un singolo valore di chiave di shard
  - Modifica della chiave di shard (resharding)

Interrogazione di shared data

- In presenza di shard, le query vengono inviate e gestite dal processo mongos, che agisce da distributore delle query e decide a quale shard devono essere inviate.
  - Proxy delle richieste
- find(): a seconda che la query includa o meno una condizione sulla chiave di shard, tale query viene rediretta solo ad un sottoinsieme degli shard interessati, oppure viene fatto un broadcast della richiesta.
- sort(): se includa la chiave di shard, mongos può determinare l'ordine con cui concatenare i risultati parziali ottenuti dai singoli shard, altrimenti ciascun shard interessato esegue un merge sort locale e poi il risultato complessivo viene ordinato da mongos.
- limit(): il limite viene imposto localmente e poi viene imposto nuovamente a livello globale da mongos

Proprietà ACID: Atomicità

- Atomicità: le operazioni all'interno di una transazione o vengono eseguite complemtanete, oppure non vengono eseguite affatto
- In MongoDB tutte le operazioni che coinvolgono un singolo documento sono sempre atomiche, anche quando le operazioni attraversano diversi sotto-documenti o array contenuti dei documenti.
- È stato stimato che l'80-90% delle applicazioni reali non richiede transazioni multi-documento.
    - Necessità di suddividere un documento in più documenti a causa del raggiungimento del limite di dimensione di 16MB per singolo documento.

Proprietà ACID: Consistenza

- Eventual consistency:
  - Garantisce che quando i dati sono stati aggiornati e propagati, tutte le letture future ritorneranno eventualmente l'ultimo stato confermato
  - Modello tipico nei sistemi distribuiti per garantire il giusto livello di prestazioni
- Strong consistency:
  - Garantisce che ogni lettura futura leggerà sempre l'ultimo valore confermato
  - Problemi di prestazioni
- MongoDB garantisce un livello di consistenza a metà tra I due modelli: bilancia il numero di possibili operazioni concorrenti da una parte, e la consistenza dei dati letti dall'altra

Proprietà ACID: Isolamento

- I vari livelli di isolamento: serializzabile, repeatable read, read committed e read uncommited, possono essere ottenuti anche in MongoDB tramite il meccanismo dei write concern, read concern e session.
- Read concern:
  - local: ritorna il dato più recente disponibile nel nodo → non è garantito che sia lo stesso dato visibile dalle altre transazioni
  - majority: garantisce che il dato letto sia quello confermato dalla maggioranza dei nodi del replica set → garantisce che le letture siano le stesse, solo se il write concern è impostato a majority, altrimenti non è garantito che il dato letto sia lo stesso per tutte le transazioni
  - snapshot: ritorna il dato presente nello snapshot della maggioranza di nodi → stessa situazione della precedente.
- Write concern:
  - majority: richiede che la maggioranza dei nodi in un replica set confermi il dato → default
  - Numero specifico di nodi in cui la scrittura deve essere propagata.

Proprietà ACID: Persistenza

- MongoDB usa il meccanismo Write-Ahead Log sul disco ogni 100 millisecond per garantire la persistenza.
- Le operazioni vengono prima scritte nel file di log e poi vengono eseguite, in questo modo è possibile ripristinare lo stato del database in caso di errori.