

ARCHITETTURA MONGODB:

Si Basa su CONCETTO di:

■ SHARDING

■ REPLICAZIONE

} Derivano dalla Natura distribuita di MongoDB

REPLICAZIONE è fondamentale per tutti i processi DISTRIBUITI.

→ Disseminare COPIE identiche di uno Stesso Data Tra Server Diversi

→ Mi SALVAGUARDA disponibilità del DATO in Caso di fallimento o Dato di un SINGOLO SERVER

Garantisce Così la RIDONDANZA , Sono più Resiliente ai guasti
■ AFFIDABILITÀ

Ci deve però ESSERE la CONSISTENZA dopo le OPERAZIONI sul DB

SHARDING Abbiamo SUDDIVISIONE di un DATASET in part. più piccole che vengono distribuite su SERVER DIVERSI

Parti ≠ su Server ≠

Vogliamo Garantire l'AFFIDABILITÀ

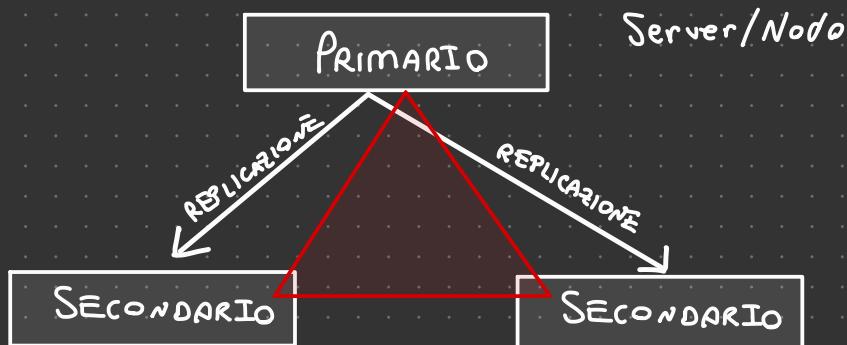
Località \Rightarrow prendo le COPIE più Vicine a me.

REPLICAZIONE:

REPLICA-SET è una Collezione di PROCESSI che vengono Chiamati Mongo Daemon che Mantengono una Coppia dello Stesso DATA-SET

Repl. mi da : $\#$ RIDONDANZA \Rightarrow Alta disponibilità da cui deriva la Tolleranza ai Guesti

Posso Aggiornare un singolo nodo senza Intervuzioni di SERVIZIO, oltre che a Migliorare le prestazioni di lettura



PRIMARIO \Rightarrow Gestisce TUTTE le Operazioni di SCRITTURA e Mantiene un Log dei Cambiamenti Intervenuti... chiamato OPLOG che poi lo hanno Anche i SECONDARI

↳ Operation

I nodi SECONDARI Usciranno per REPLICARE le operazioni scritte NELL'OPLOG del Nodo PRIMARIO.

Ogni REPLICASET ha Solo un Nodo primario

SE Nodo PRIMARIO DIVENTA Non DISPONIBILE allora È
Necessario che un Nodo SECONDARIO Venga ELETTO come Nodo
primario

L'elezione ha un PROTOCOLLO, Basata su RAFT.

→ Meccanismo di VOTO (Per eleggere nuovo nodo PRIMARIO)

Algoritmo di ELEZIONE viene Attivato Per

■ Perdita Nodo Primario \Rightarrow Errore Comunicazione Con nodi SEC.
(se dopo TIMEOUT il Primario non risponde più)

■ Aggiunta/Rimozione Nodo dal REPLICASET

■ Inizializza REPLICASET

Votazione deve Richiedere minor Tempo possibile (Al più Solitamente
12 Secondi per Completare, compreso TIMEOUT di 10s)

IL NODO SECONDARIO con TIMESTAMP di Scrittura più RECENTE
è Quello con Maggiore probabilità di diventare primario (voglio
MINIMIZZARE #ROLLBACK DA FARE)

Voglio prendere il Nodo più AGGIORNATO.

Sino a Che PROC. DI ELEZIONE non è Terminato non si possono Effettuare Scritture

Se ci Sono micro-INTERRUZIONI Continue allora potrei avere delle ELEZIONI CONTINUE CONSECUTIVE

Dopo un ELEZIONE i Nodi entrano in Una fase di Congelam. Nella Quale non possono fare più ELEZIONI per EVITARE di DESTABILIZZARE il SISTEMA.

STABILITÀ DEL SISTEMA Si BASA SULL' OpLog, dove ci Sono tutte le OPERAZIONI EFFETTUATE.

↳ Operazioni, Come Già Seppiemo Sono IDEMPOTENTI

Rieseguire le OPERAZIONI + Volte non cambierà

REPLICA-SET By default prevede 3 Nodi Membri.

↳ #Nodi deve Essere dispari da BEST-PRACTICES, Per Evitare Situazioni di STALLO Nelle VOTAZIONI

Se ho #Nodi Pari mi Serve un ARBITRO, Ovvero
↳ Per PARITÀ Nei VOTI

Ci Possono Essere dei MEMBRI Nascosti/Ricordati ai Quali Viene Affidata dei TASK Particolari come BACK-UP o REPORTING

Assicurarsi che Almeno un MEMBRO del REPLICA-SET Si Trovi in un DATA-CENTER Alternativo

ARBITRO è un NODO che non ha la copia del DATASET, e non può diventare primario.

Può stare su una Macchina più leggera e può SOLAMENTE Votare.

NODI NASCOSTI non possono diventare PRIMARI, Non ci posso Accedere con le APPLICAZIONI Client ma Possono Partecipare Alle Votazioni

Ritardarli significa Che posso avere uno SNAPSHOT delle BASE DI DATI

SCRITTURA SUL NODO PRIMARIO e POI REPLICATE SUI SECONDARI

è ASINCRONA, non mi preoccupa della Scrittura sui SECONDARI

Devo Capire Se le Scritture sono andate a Buon Fine o Meno

Devo CERTIFICARE Che le Scritture siano "DEFINITIVA" Quando l'ho postata su un CERTO NUMERO di NODI

→ dipende dalle CONFIGURAZIONI → WRITE CONCERN

3 Parametri:

W = # Nodi Replica-Set che devono aver Confermato la SCRITTURA prima che Tale OPERAZIONE sia considerata Conclusa

Veloci possibili Sono:

- 1 = Solo Nodo primario
- Majority
- Numero che Imposta io

Più aumento il Numero e più Sarà la CATENZA ma anche la PERSISTENZA

J = Se si Richiede che l'OPERAZIONE Sia State Scritte su Disco (+ Sicurezza, - Prestazioni)

W Timeout = limite Tempo per OPERAZIONE DI SCRITTURA

Se NODO PRIMARIO Non Risponde Allora WRITE Si Blocca

LETTURE:

MongoDB Redige sul nodo Primario ma non è NECESSARIO.

Posso decidere di distribuirle Su più Nodi Sfruttando il PRINCIPIO DI LOCALITÀ.

- Primary
- Primary Preferred
- Secondary
- Sec. Pref.
- Nearest

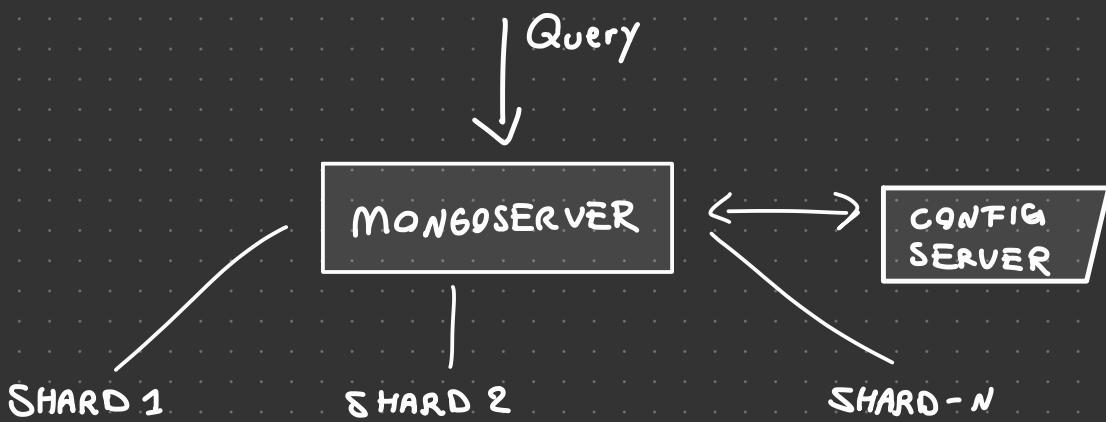
Per Garantire ISOLAMENTO e CONSISTENZA posso SPECIFICARE il READ CONCERN

- ① **Local** \Rightarrow Ritorno il DATO disponibile Nel NODO Locale al Momento della QUERY
- ② **Available** \Rightarrow Ritorno il DATO più Aggiornato DISPONIBILE più Vicino
- ③ **Majority** \Rightarrow Dato CONFERMATO dalla Maggioranza
- ④ **Linearizable** \Rightarrow Dato Confermato da TUTTO il Replica-Set
- ⑤ **Snap Shot** \Rightarrow Dato Confermato dalla Maggioranza in un Certo Istante di TEMPO.

SHARDING:

Possibilità di Suddividere in piccole Parti e Separarli.

- Shard \Rightarrow 1 Replica-Set che MEMORIZZA una Porzione di DATA-SET
- MongoServer \Rightarrow Distributore delle QUERY
- ConfigServer Contiene I METADATI Sulle Varie Parti



Sharding Implementato A Col di Collezione

VANTAGGI:

■ Aumento Prestazioni (R/W) \Rightarrow Parallelismo

■ Aumento STORAGE Complessivo

■ Sfruttare le LOCALITÀ del DATO

SHARD-KEY \Rightarrow Una o più Proprietà dei DOCUMENTI



Se ho valori più probabili degli
altri ha uno sbilanciamento che
mi crea dei coni di bottiglia

BALANCER è un COMPONENTE che periodicamente bilancia
gli SHARD

MONGODB e le PROPRIETÀ ACID:

ATOMICITÀ, tutte le op. su un solo DOCUMENTO sono ATOMICHE
CONSISTENZA, non esiste, ma invece ci sono:

- EVENTUAL Consistency (dato sarà propagato allora le letture future riporteranno l'ultimo)
- STRONG Consistency (lul Marx.) stato confermato

PERSISTENZA, Garantita attraverso WRITE-AHEAD LOG

ISOLAMENTO, livelli di ISOLAMENTO li raggiungo attraverso apposite Configurazioni di WRITE CONCERN e READ CONCERN

Majority + Majority

Massimo livello di ISOLAMENTO