|  |
| --- |
| Relazione Progetto Finale del corso di Big Data |
| Gruppo Miel Pops |
|  |
| A cura di Gaetano Bonofiglio e Veronica Iovinella |
|  |

Relazione Progetto Finale del corso di Big Data

Gruppo Miel Pops

# Il progetto: SQL vs NoSQL vs NewSQL

Il progetto svolto riguarda l’analisi di diversi DBMS, in particolare si concentra sul confronto tra le tecnologie SQL, NoSQL e le emergenti NewSQL.

Con l’avvento dei Big Data si sono poste nuove sfide da affrontare per quanto riguarda le sostenibilità delle performance al crescere della base di dati. L’enorme mole di dati e la richiesta intensiva di operazioni al secondo ha portato alla naturale evoluzione dei classici server verso una scalabilità orizzontale, portando alla nascita dei cluster di calcolatori e la costruzione di data center dedicati.

Come si pone la tecnologia NewSQL in questo contesto? I classici database relazioni continuano ad avere una certa attrattiva per via del modello, adattissimo alla rappresentazione a oggetti, e le proprietà ACID delle transazioni garantite che restano una necessità per diverse applicazioni di casi d’uso reali.   
L’idea alla base dei sistemi NewSQL è quella di portare il classico database relazionale su cluster, sfruttando i nodi non solo per la ridondanza dei dati, ma dividendo effettivamente il carico e il dataset in blocchi.  
Gli approcci possibili sono diversi, e vanno da architetture shared-nothing completamente nuove, ad ottimizzazioni di engine già esistenti per il trasporto su cluster.

Trovando il database frammentato tra i diversi nodi, le possibili problematiche riguardo ad operazioni tipicamente relazionali (i.e. join) restano teoricamente le stesse dei già ben noti sistemi NoSQL.

Il nostro esperimento si svolgerà quindi nel seguente modo:

* Proponiamo un’analisi dei sistemi presi in esame, verificando le proprietà delle architetture.
* Verifica delle prestazioni su operazioni quali:
  + Inserimenti
  + Ricerca non indicizzata (ove possibile)
  + Ricerca indicizzata
  + Aggregazioni (ove possibile)
  + Join (ove possibile)
  + Throughput di operazioni (letture) al secondo
* Le operazioni di cui sopra verranno eseguite:
  + Su nodo singolo
  + Su cluster (ove possibile)

I sistemi presi in esame sono:

* Sistemi **SQL**:
  + Postgres
  + MySQL
* Sistemi **NoSQL**:
  + MongoDB
  + Redis
  + Neo4j
  + OrientDB
  + Cassandra
* Sistemi **NewSQL**:
  + MemSQL
  + Voltdb
  + CockRoach
  + Trafodion
  + NuoDB
  + Clustrix

Per eseguire un’analisi precisa delle prestazioni, utilizzeremo come piattaforma **Docker**, avviando ogni sistema in un container, in modo da dedicare le stesse risorse di calcolo a tutti i sistemi.

# Architetture dei DBMS

## Sistemi SQL

### Postgres

Postgres è un classico DBMS relazionale. Utilizza un’architettura client/server ed un sistema di transazioni su connessione TCP/IP per l’esecuzione delle operazioni.   
Le operazioni del nostro esperimento sono tutte supportate, tuttavia non esiste una versione in cluster di tale sistema, per cui non varrà preso in esame per quel frangente.

### MySQL

MySQL è il secondo DMBS relazionale che prendiamo in esame. L’architettura è sempre di tipo client/server, e le operazioni sono chiaramente tutte supportate.

A differenza di Postgres, esiste una versione cluster di MySQL. L’architettura è di tipo shared-nothing, per cui ogni nodo opera in maniera indipendente.  
I nodi si distinguono tra SQL nodes, che eseguono MySQL server, e i Data Nodes, che contendono diversi blocchi del dataset. Questa divisione permette ad ogni SQL node di avere accesso in ogni momento all’intero database, garantendo le proprietà delle transazioni.  
Questo tipo di architettura inoltre permette di non avere single point of failure, in quando ogni nodo resta indipendente, mentre per aumentare la robustezza basta introdurre un fattore di replicazione dei data nodes.

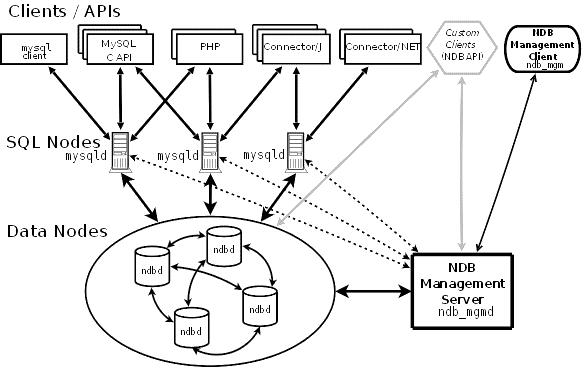


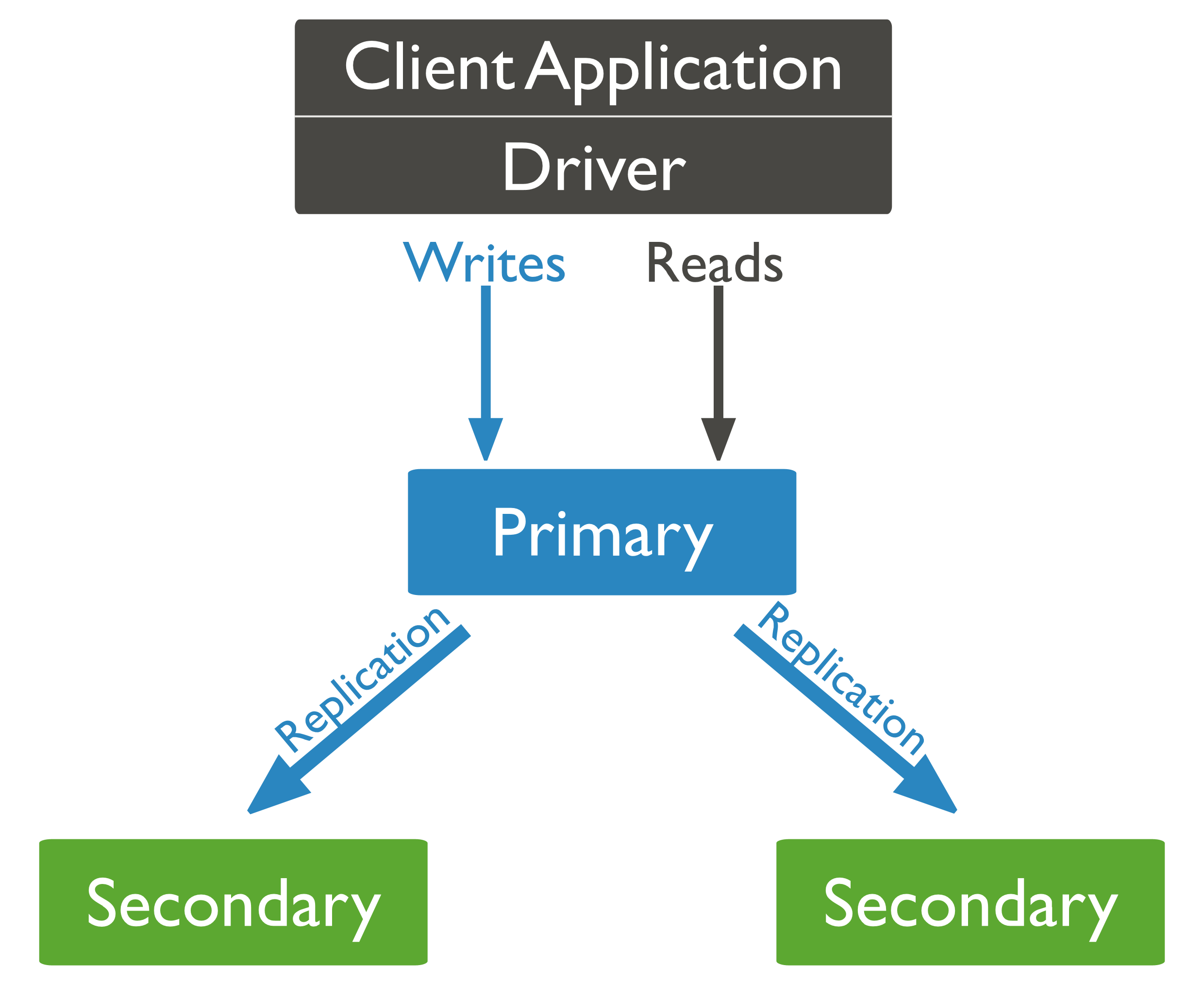
Figura Architettura di MySQL Cluster

## Sistemi NoSQL

### MongoDB

MongoDB è un document-store database che permette di esprimere attraverso la definizione dello schema anche relazioni tra oggetti, offrendo una gran varietà di query possibili per interrogare i documenti. Data la sua natura, non supporta tuttavia le operazioni di join (eccetto con un’operazione di lookup), alle quali si può tuttavia ovviare con una apposita definizione dello schema e inserendo un certo grado di ridondanza. Le funzioni di aggregazione sono tuttavia implementate, attraverso il paradigma MapReduce.

Mongo ha la possibilità di scalare orizzontalmente, dividendo il dataset in shard secondo un’arhcitettura master/slave. Prevede inoltre la possibilità di indicare un Replica set, ovvero un fattore di replicazione dei dati.



Nonostante sia un sistema NoSQL, Mongo ammette un certo livello di transazioni ACID, ovvero a livello di documento.

### Redis

Redis è un particolarissimo database in-memory che opera su coppie key-value. Permette all’occorrenza di persistere i dati in memoria secondaria.

Per via della particolare architettura, Redis non ammette parecchie operazioni che sono invece tipiche degli altri sistemi: tra queste la ricerca non indicizzata, alla quale si può ovviare solo aggiungendo un livello di indicizzazione sui values, e le operazioni di join, poiché si tratta di un’operazione che coinvolge i values.

Esiste una versione cluster di Redis, che può scalare fino a 1000 nodi, ma porta con sé una forte limitazione perché non supporta le operazioni multi-key, ma solo le operazioni single-key. Ciò è dovuto all’architettura in-memory del DBMS.

La replicazione è ottenuta attraverso l’architettura master/slave con un meccanismo publisher/subscriber, e può essere sviluppata su più livelli, permettendo ad uno slave di essere master di un altro slave.

Nonostante le limitazioni, Redis trova ampio uso in particolari casi d’uso, ad esempio implementazioni di code o caching, e proprio grazie a queste limitazioni offre prestazioni elevatissime.