
cassandra

Storia

Nasce inizialmente all'interno di Facebook per gestire le ricerche fra i messaggi.
Attualmente open-source, è uno dei database NoSQL più diffusi.
E' incluso tra i database NOSQL Column family perché ha un datamodel ispirato a BigTable.
Molte aziende hanno implementato con successo e beneficiato di Apache Cassandra tra cui alcune grandi aziende come: di Apple , Comcast , Instagram , Spotify , eBay , Rackspace , Netflix , e molti altri. Gli ambienti di produzione più grandi hanno PB di dati in gruppi di oltre 75.000 nodi. Cassandra è disponibile sotto la licenza Apache 3.0.

Caratteristiche 1/2

Cassandra è una database distribuito, fault-tolerant, elastico e con consistenza dei dati regolabile sia in read che in write; questo significa che il server è solitamente installato in una configurazione clustered nella quale più nodi di Cassandra cooperano per ottimizzare e distribuire le informazioni.

Nessun nodo è in alcun modo diverso dagli altri, in questo modo non esiste all'interno del cluster una specificità critica che in caso di malfunzionamento possa rendere inoperante il database.

Caratteristiche 2/2

I dati sono automaticamente duplicati su più nodi, questo garantisce che all'eventuale crash di un elemento dell'insieme non debba necessariamente seguitare la perdita di informazioni importanti o lo stallo dell'intera istanza di Cassandra.

Struttura

Il modello dati prevede un'organizzazione che parte dal cluster come entità di più alto livello e scende fino alla singola colonna. Gli elementi che ne fanno parte sono i seguenti:

- Cluster
- Keyspace
- Column family
- Column
- Supercolumn
- Row

Cluster

L'insieme dei server che costituiscono l'istanza di cassandra. Il cluster può contenere una o più entità di livello inferiore, i keyspace.

Keyspace

Un namespace per le column family; può essere assimilato allo schema oppure al database di un RDBMS.

Column family

È un contenitore di colonne (o supercolumn). La column family è l'equivalente della tabella in un RDBMS. Ogni column family è salvata in un file separato e i valori sono ordinati per chiave della riga (row key). Le column family possono essere statiche o dinamiche. Nel primo caso i metadati delle colonne (per esempio il nome e il tipo di dato) sono definiti a priori, mentre nel secondo caso sono definiti al momento dell'inserimento della colonna da parte dell'applicazione.

Column

La colonna è definita attraverso un nome e, per ciascuna riga, contiene un valore e un timestamp. Quest'ultimo è utilizzato per la risoluzione dei conflitti. Visto che ciascuna column family è salvata su un file separato, è consigliabile raggruppare nella stessa column family le colonne a cui spesso si accede contemporaneamente. Un singolo valore non può essere più grande di 2 GB e il numero massimo di colonne per una certa riga non può superare i due miliardi.

Supercolumn

La supercolumn è una colonna che a sua volta contiene altre colonne.

Row

La riga è identificata da una chiave, la row key (che non può essere più grande di 64 KB), e contiene valori appartenenti a più colonne. Ciascuna riga di una colonna family può contenere valori di tutte le colonne o solo di alcune. La chiave determina su quale server sono salvati i dati. Una riga deve poter essere contenuta in un singolo server, quindi la dimensione della riga ha come limite lo spazio disco disponibile.

Installazione: requisiti

- Ultima versione di Oracle Java Platform, Standard Edition 8 (JDK) è raccomandato o OpenJDK 7.
- Python 2.6+ (necessario se si installa OpsCenter).

Installazione su qualsiasi piattaforma basata su Linux

- **Verifica della versione java (Java 8):**

  $ java -version

- **Download di DataStax Distribution di Apache Cassandra 3.x:**

  $ curl -L http://downloads.datastax.com/datastax-ddc/datastax-ddc-version_number-bin.tar.gz | tar xz

- **Per configurare, passare alla directory di install / conf:**

  $ cd datastax-ddc-version_number/conf

CQL

- **A differenza di molti altri database NoSQL Cassandra dispone di un proprio linguaggio di query: CQL**
- **L'assonanza nel nome vuole richiamare le forti similitudini con SQL.**
- **Il CQL può essere utilizzato via API o attraverso una shell cqlsh**

CQL – Creare un DB

Semplificando, un DB è una collezione di tabelle; l'equivalente in Cassandra prende il nome di keyspace, che inoltre definisce anche una strategia di replica dei dati:

Esistono due strategie a seconda che si utilizzi un solo cluster (SimpleStrategy) o più cluster (NetworkTopologyStrategy). La sintassi per creare un keyspace è la seguente:

```
CREATE KEYSPACE [keyspace_name]
WITH REPLICATION = {'class' : 'Strategy', [datacenter_name] : n };
```

Dove keyspace_name è il nome che si vuole dare al keyspace ed n indica il numero di repliche di dati che deve essere presente nel cluster.

CQL – Creare un DB

Nel caso di un solo datacenter non è necessario assegnargli un nome e la stringa associata

CREATE KEYSPACE keyspace_name WITH REPLICATION = {'class' : 'SimpleStrategy', 'replication_factor' : n };

Nel caso di più datacenter è necessario specificare il nome del datacenter ed il replica_factor per ciascuno di essi:

CREATE KEYSPACE keyspace_name WITH REPLICATION = {'class' : 'NetworkTopologyStrategy', 'dataprim' : 3 , 'datas' : 2 };

CQL – Lavorare con un Keyspace

E' possibile selezionare un keyspace per lavorare solo con le tabelle di un particolare DB:

USE keyspace_name;

oppure modificarlo, ad esempio per cambiare il replication factor od aggiungere datacenter, attraverso il comando ALTER KEYSPACE che segue la sintassi di

CREATE KEYSPACE.

Inoltre è possibile ottenre l'elenco di tutti i keyspace con il comando

DESCRIBE KEYSPACES;

o alternativamente interrogando la tabella system(che contiene i metadati):

SELECT * FROM system.schema_keyspaces;

CQL – Creare una Tabella

La sintassi per la creazione di tabelle (ovvero le column family) è la seguente:

CREATE TABLE keyspace_name.table_name

(column_definition, column_definition, ...)

WITH property AND property .

Il keyspace_name può essere omesso se si deve creare una tabella nel keyspace attualmente in uso; e column_definition può seguire uno dei seguenti formati :

column name cql_type

column name cql_type PRIMARY KEY

PRIMARY KEY (partition key)

column name collection_type

CQL – Inserimento Dati

Per inserire dei dati all'interno delle tabelle dobbiamo utilizzare il comando INSERT:

INSERT INTO keyspace_name.table_name

(column_name, column_name...)

VALUES (value, value ...)

USING option AND option

I valori devono rispettare la tipologia specificata per la colonna, se si tratta di dati testuali devono essere racchiusi fra apici.

Una operazione di INSERT inserisce una o più colonne in una tabella di Cassandra in maniera atomica ed isolata (nel senso delle proprietà ACID)

CQL – Inserimento Dati

Per inserire dati nelle collection i valori devono essere specificati all'interno di parentesi [] nel caso di liste, {} di set e mappe.

LIST→[value, value ...]

SET→{value, value ...}

MAP→{value_key : value, value_key : value ...}

E' bene precisare che in CQL:

- **le listservono per memorizzare valori univoci e ordinati; i valori possono essere inseriti in qualunque ordine e verranno restituiti secondo il loro ordinamento naturale (es. alfabetico);**
- **i set consentono di inserire duplicati ed i valori verranno restituiti secondo l'ordine di inserimento;**

CQL – Ricerca e selezione dei Dati

Per interrogare i dati si utilizza il comando SELECT.

SELECT selector

FROM keyspace_name.table_name

WHERE relation AND relation ...

ORDER BY (clustering_column (ASC | DESC)...)

LIMIT n

ALLOW FILTERING