UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione

Laurea Magistrale di Ingegneria Informatica

Corso di Big Data Analytics and Business Intelligence

Expert Finding

Identificazione utenti con conoscenza su un dato tema

Autori:
Davide Lauretano
M63000792
Michele Pommella
M63000790
Davide Trimaldi
M63000799

Professore:
Antonio Picariello

Anno Accademico 2018-2019

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Problema

La ricerca di esperti è una sfida per molte ragioni, tra cui:

- Il volume di pubblicazioni di un esperto non è indice di esperienza;
- Il primo esperto che hai trovato potrebbe non essere il migliore;
- Alcuni argomenti generano più opinioni che fatti e quindi trovare il vero esperto può essere difficile;
- Di solito c'è una mancanza di accesso alle informazioni sulle prestazioni passate dell' esperto;
- Le competenze non sono distribuite in modo uniforme e punti di forza delle associazioni tra esperti variare in modo significativo
- Non ci sono standard che specificano i criteri e/o le qualifiche necessarie per particolari livelli di esperienza.
- La vera esperienza è rara e costosa. Spesso l'accesso è controllato, in modo informale o formalmente, dall'esperto stesso o dal loro management.
- La competenza di un esperto cambia continuamente e richiede consapevolezza di questa dinamica.
- Le soluzioni a problemi complessi spesso richiedono o comunità di esperti o diverse gamme di competenze che devono essere riunite per risolvere il complesso i problemi.
- Gli ingegneri di uno studio classico hanno trascorso il 16% del loro tempo a comunicare esperti - ma la comunicazione è stata ostacolata da differenze geografiche, di fuso orario e barriere culturali.

In sintesi, la ricerca di esperti è un compito complesso e difficile.

Capitolo 2

Metodologia ed architettura

La metodologia adoperata fa uso di un database *NoSQL*, per il mantenimento dei dati, e della tecnologia *Apache Spark* su macchina virtuale *Azure*.

La macchina virtuale mette a disponizione:

- 8 core
- Memoria centrale di 58 GB
- Memoria di massa di 30 GB
- Disco temporaneo secondario di 400 GB

Accessibile mediante protocollo SSH, offre l'utilizzo dell'engine Apache Spark con stack relativo già installato e configurato. In particolare si è fatto uso del modello di programmazione Spark per il linguaggio Python: *Pyspark*.

Il database NoSQL utilizzato è *MongoDB* per la tipologia di dati da trattare e le operazioni da effettuare su di essi. MongoDB è un database documentale che consente di trattare aggregati strutturati, risultando in accessi più flessibili. Permette, quindi, di sottomettere query basate sui campi dell'aggregato e recuperare solo parte di esso. MongoDB memorizza e restituisce documenti in formato *BSON*, rappresentazione binaria del JSON. I documenti sono strutture dati ad albero gerarchiche e possono essere anche strutturalmente non identici. Sono infatti distinti per collezione, insieme di documenti simili. Le caratteristiche di MongoDB hanno consentito una gestione semplice ed efficace del dataset composto principalmente da file *JSON* e *XML*. Il flessibile modello documentale dei dati con schema dinamico e ridimensionamento automatico su hardware comune rende MongoDB ideale per applicazioni con grandi volumi di dati multi-strutturati e dall'elevato tasso di cambiamento. Inoltre MongoDB offre la completa interoperabilità

con il sistema Spark, mediante il *MongoDB Spark Connector*, che ne consente la gestione tramite codice Pyspark.

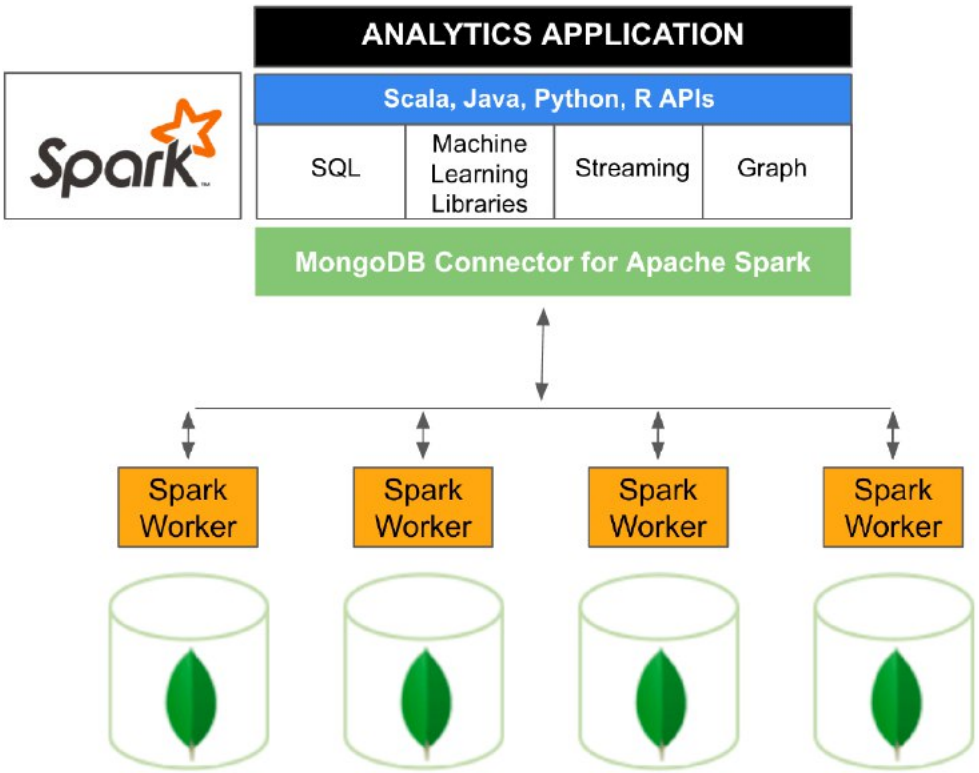

Figura 2.1: MongoDB Spark Connector

Apache Spark è un framework di data processing che consente di effettuare query veloci grazie alla memorizzazione *in-memory* dei dati. Supporta diversi linguaggi di programmazione: Scala, Java, R, Python. Le sue principali caratteristiche sono:

Velocità sfruttando le ottimizzazioni in-memory;

Framework unificato offrendo packages di librerie di alto livello (supporto a query SQL, Machine Learning, stream e graph processing);

Semplicità includendo API facili da usare per operare su grandi dataset, come operatori per trasformare e manipolare dati semistrutturati.

Spark, combinato con Python, è stato sfruttato sia per il caricamento del dataset di file XML nel database, non nativamente convertibile in BSON da MongoDB, sia per l'elaborazione dei dati.

L'unione di queste due tecnologie consente allo sviluppatore di realizzare l'applicazione più velocemente, utilizzando un solo database. Spark può eseguire direttamente sui dati operativi posti in MongoDB, senza tempi e costi di un processo di ETL. MongoDB può efficientemente presentare di ritorno i risultati analitici ai processi operativi.

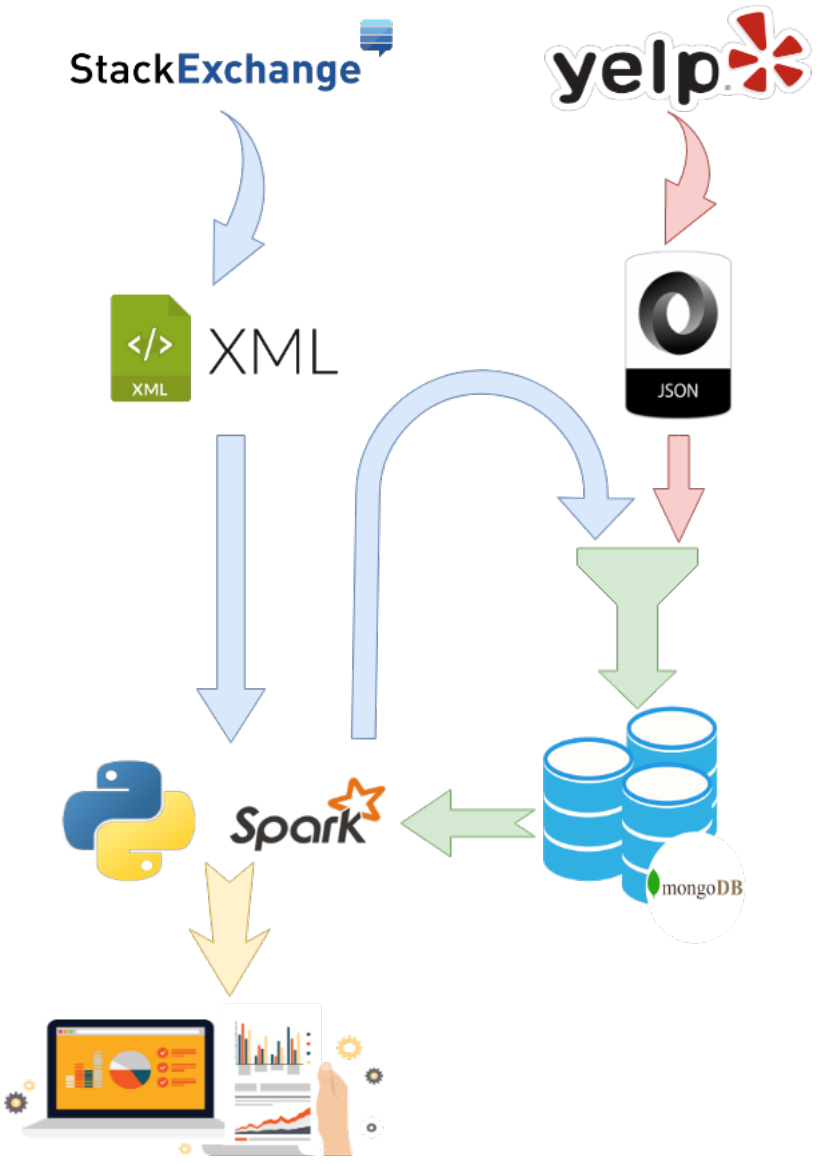

Figura 2.2: Architettura

Capitolo 3

Risultati sperimentali