U-SQL - Un nou limbaj care îmbunătățește procesarea Big Data

Iordăchescu Anca-Mihaela, grupa 407

Analizând caracteristicile analiticelor Big Data, se poate observa faptul că limbajele de programare care au ca specific manipularea Big Data trebuie să respecte anumite aspecte astfel încât să se asigure servicii ușor de folosit și eficiente. În primul rând acestea trebuie să permită procesarea oricărui tip de date. De la a analiza modelele atacurilor de tip BotNet din log-urile de securitate, până la procesarea imaginilor și videoclipurilor folosind machine learning, limbajul respectiv trebuie să îți permită să lucrezi cu orice tip de date. În al doilea, trebuie ca utilizatorului să i se ofere posibilitatea de a își customiza codul, întrucât unii algoritmi mai complecși necesită procesări custom, precum diferite funcții definite de utilizator și formate de input/output specifice problemei adresate, ce nu sunt posibil de realizat în limbajele de interogare standard. Iar în al treilea rând aceste limbaje trebuie să asigure o scalare eficientă a volumului de date, fără a trebui ca utilizatorul să folosească diferite topologii de scalare sau limitări.

Limbajele SQL, precum Hive, vin cu o abordare declarativă care asigură scalarea datelor, executarea în paralel a comenzilor și operațiile de optimizare, făcându-le astfel ușor de utilizat de programatori și asigurând totodată eficiență în lucrul cu diferite analitici și warehouse-uri. Însă, limbajele de acest tip au limitări în ce privește extensibilitatea modelelor și suportul pentru tipurile de date nestructurate, iar prelucrarea unor asemenea fișiere este îngreunată considerabil. De exemplu, dacă se dorește manipularea anumitor fișiere cu date sau a unor surse externe, este nevoie mai întâi de crearea unor obiecte care să schematizeze datele respective înainte de a putea fi folosite în efectuarea diferitelor cerințe. Chiar daca limbajele SQL au capabilități de formatare custom și de utilizare de funcții definite de utilizator, acestea sunt mai greu de implementat, de integrat și implicit de realizat mentenanță asupra lor.

Metodele de procesare a Big Data folosind limbaje de programare, asigură o modalitate ușoară de a scrie cod custom. Programatorul trebuie să scrie cod pentru scalarea și asigurarea performanței comenzilor, cod însă ce poate fi greu de scris corect și optim pentru a asigura eficiență. O altă problemă e faptul că SQL poate fi integrat ca string și din cauza

faptului că nu există tool-uri care să asigure suport, posibilitățile de extensibilitate sunt limitate, iar din cauza scrierii de cod procedural, optimizarea este greu de realizat [1].

Astfel, dorindu-se rezolvarea atât a problemelor aduse de limbajele procedurale, cât și a celor aduse de limbajele declarative, SQL, a apărut U-SQL, care combină sintaxa declarativă SQL cu limbajul de programare procedural C#. U-SQL asigură atât procesarea datelor structurate, cât și nestructurate, dar și a surselor externe, fiind totodată posibilă extensibilitatea și optimizarea programelor scrise, utilizând diferite funcții custom scrise în C#.

La baza U-SQL stă SCOPE, un limbaj de programare declarativ creat de Microsoft în 2008 pentru a permite analiștilor familiarizați cu SQL să realizeze operații pe baze de date foarte mari, care erau păstrate în platforme distribuite, precum Hadoop, în loc de baze de date relaționale. U-SQL este foarte asemănător cu Pig, Hive și Spark, aducând în prim-plan concepte specifice procesării Big Data, precum schema on read, lazy evaluation și custom processors and reducers [2].

Există trei pași ce sunt urmați când se face procesarea Big Data folosind U-SQL. În primul rând are loc extragerea datelor folosind EXTRACT, pentru fiecare informație extrasă atribuindu-se un tip de date specific C#. Această operație poate fi realizată atât din fișiere nestructurate și semistructurate precum JSON și XML, cât și din surse structurate sub formă de tabele. Urmează transformarea datelor folosind SQL sau diferite funcții scrise de programator. Script-urile U-SQL utilizează atât tipuri de date, operatori și expresii specifice C#, cât și operații specifice SQL. Câteva exemple sunt operațiile de tip LDD (Limbajul de Definire al Datelor) și LMD (Limbajul de Modificare al Datelor). De asemenea, cuvintele cheie folosite în scrierea query-urilor, SELECT, WHERE și JOIN sunt și ele întâlnite, doar că pentru a se evita erorile de sintaxă trebuie scrise cu majuscule, aspect contrar SQL. Totodată, este posibilă utilizarea construcțiilor de forma IF…ELSE, însă nu și a celor de tipul WHILE și FOR. Iar la final se execută operația de constituire a output-ului, care poate fi atât în format nestructurat (.csv, .tsv), cât și sub forma unui tabel U-SQL, pentru a putea fi supus unor alte operații ulterioare.

Sunt întâlnite două metode de a rula un script U-SQL, fie este executat local, folosind Visual Studio, fiind însă mai întâi necesară pregătirea mediului de lucru prin instalarea tool-ului Azure Data Lake and Stream Analytics Tools, fie în cloud folosind Azure Data Lake. În cazul primei situații, datele de input și output și script-ul sunt salvate local, în timp ce în al doilea caz, acestea utilizează resurse ale Azure Cloud, lucru ce implică anumite costuri pentru folosirea și stocarea resurselor. Din acest motiv, în stagiile incipite ale unui

proiect ce presupune manipulare de date folosind U-SQL este recomandată dezvoltarea inițială la nivel local, folosind Visual Studio, pentru a se putea evita cheltuielile impuse de serviciile de cloud [3].

În imaginile următoare voi evidenția un script U-SQL trecut prin toate etapele descrise anterior.

În primul rând am declarat variabilele în care voi salva fișierele de input și output, iar apoi, folosind EXTRACT am parcurs și salvat într-o nouă variabilă datele .csv-ului atașat, atribuind tipurile de date corespunzătoare. Întrucât este vorba de un fișier .csv, a trebuit să ignor primul rând deoarece acesta conține header-ul.

```
1  // Script - RestaurantScript.usql
2  // Variables for input and output file name and paths
3  DECLARE @inputFile = "/input/restaurants_ratings.csv";
4  DECLARE @outputFile = "/output/restaurants_out.csv";
5
6  // Data is extracted from input file (CSV) and stored in employees rowset variable
7  @restaurant_ratings =
8      EXTRACT rest_id int,
9              name string,
10             address string,
11             online_order bool,
12             book_order bool,
13             rate double,
14             votes int,
15             phone string,
16             location string,
17             rest_type string,
18             favorite_dish_id int
19     FROM @inputFile
20     USING Extractors.Csv(skipFirstNRows:1); // Skip first row which contain headers
```

În următoarea fază am prelucrat datele furnizate și am scris un query care să îmi returneze pentru restaurantele care au rating minim 4 și asigură serviciu de comandă online un set de informații. Pentru a evidenția și interacțiunea U-SQL - C#, am scris și o funcție într-un script C# pe care o apelez în query-ul descris anterior (rândul 24).

```
22 // Transformation Step: Columns are renamed and no of rows are filtered
23 @bestOnlineRestaurants =
24     SELECT USQLApp1.Helpers.FormatRestaurantName(name, location, rest_type) AS Name,
25             rate AS Rating,
26             online_order AS OnlineOrder,
27             phone AS Phone,
28             favorite_dish AS FavoriteDish
29     FROM @restaurant_ratings
30     WHERE rate > 4 && online_order == true;
```

```
namespace UsqlApp1
{
    public static class Helpers
    {
        public static string FormatRestaurantName(string name, string location, string restaurantType)
        {
            return name + " (" + restaurantType + ") - " + location;
   // Note that U-SQL does not yet support new C# 7.0 string interpolation
            // return $"{name} ( {restaurantType} ) - {location}";
        }
    }
}
```

Iar la final preiau rezultatul query-ului de la etapă precedentă și îl inserez în fișierul creat:

```
// Load transformed data to output file
OUTPUT @bestOnlineRestaurants
TO @outputFile
USING Outputters.Csv(outputHeader:true); // Write column names/headers to output file
```

Concluzionând, se poate afirma faptul ca U-SQL reprezintă un limbaj de programare care aduce beneficii foarte mari în ceea ce privește manipularea și procesarea Big Data, deoarece combinând sintaxa declarativă SQL cu limbajul de programare C#, asigură atât procesarea datelor structurate, cât și a celor nestructurate și care provin din surse externe. Un alte aspect important e posibilitatea de customizare a codului prin utilizarea unor funcții scrise în C# ce pot fi mai pe urmă preluate și utilizate în script-urile U-SQL, asigurându-se extensibilitatea și optimizarea programelor scrise.

REFERINȚE

[1]https://devblogs.microsoft.com/visualstudio/introducing-u-sql-a-language-that-makes-big-data-processing-easy/

[2] https://www.techtarget.com/searchdatamanagement/definition/U-SQL

[3]https://www.red-gate.com/simple-talk/cloud/azure/processing-data-using-azure-data-lake-azure-data-analytics-and-microsofts-big-data-language-u-sql/