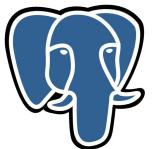




Modellazione ed Architetture Avanzate di Database

Simone Cullino & Roger Ferrod

Analisi della polarità di recensioni di prodotti Amazon



PostgreSQL



mongoDB



1. INTRODUZIONE

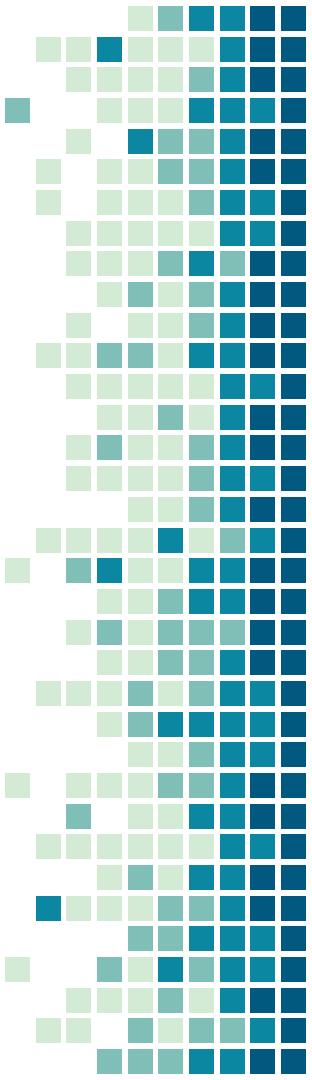


INTRODUZIONE



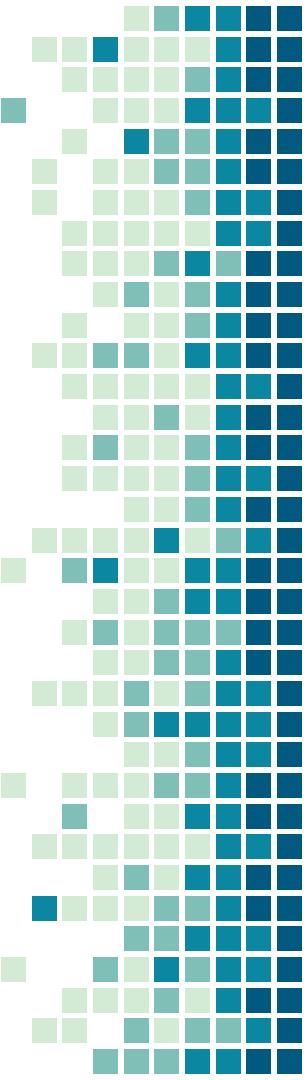
Lo scopo del progetto è quello di:

- Utilizzare un DB NoSQL (MongoDB)
- Utilizzare un DB Relazionale SQL (Postgres)
- Analizzare la polarità di recensioni di prodotti di Amazon e confrontarle con il numero di stelle attribuite (Gold Value) usando un indice di correlazione (Pearson)
- Confrontare le performance delle soluzioni DBMS adottate



2. ANALISI DELLA POLARITA' - VADER





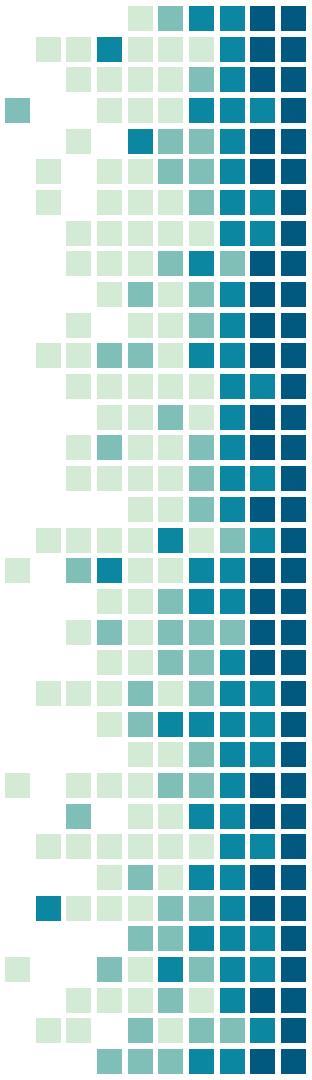
ANALISI DELLA POLARITÀ - VADER

Abbiamo deciso di utilizzare, come algoritmo di analisi della polarità **VADER**, un metodo proposto da *C.J Hutto*.

L'algoritmo attribuisce ad ogni frase un valore numerico racchiuso in un intervallo $<-1,1>$

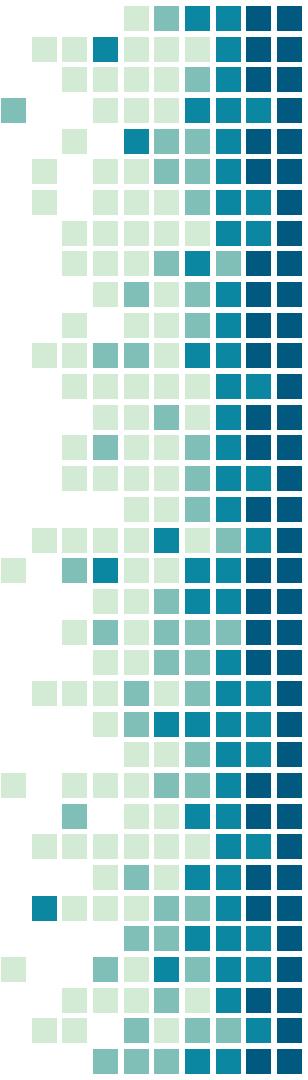
Media aritmetica tra score del titolo e score del corpo della recensione

ANALISI DELLA POLARITA' - VADER



Le risorse lessicali utilizzate, fornite sia dall'autore che dal corso, sono:

- ❑ **Lexicon.txt**: 7517 parole ed emoticons annotate manualmente con i relativi punteggi di polarità
- ❑ **Modifiers.txt**: 66 parole annotate, tra avverbi e aggettivi che modificano la polarità (ex little)
- ❑ **Negate.txt**: 58 forme di negazione
- ❑ **Slang.txt** : 61 slangs con descrizione
- ❑ **Specials.txt** : 5 modi di dire annotati (ex: yeah right)

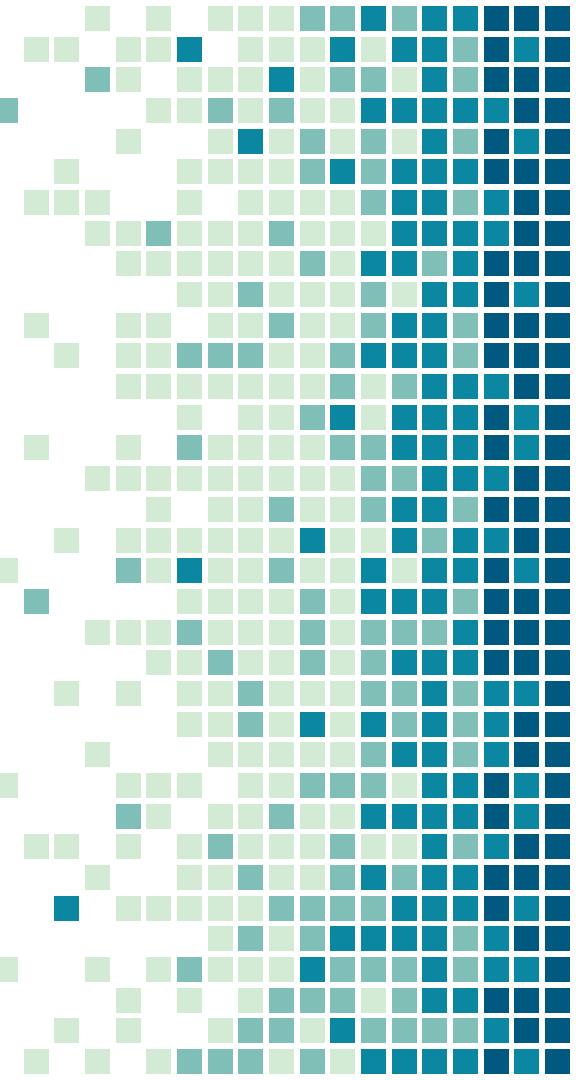


ANALISI DELLA POLARITÀ - VADER

L'algoritmo presenta 5 euristiche che modificano (con un fattore di scala empirico) il valore di polarità della frase:

- ❑ **Punteggiatura:** incrementa l'intensità senza modificare la semantica (ES: this is great!!!)
- ❑ **Lettere maiuscole:** una parola tutta maiuscola ha più importanza (ES: very USEFUL)
- ❑ **Modificatori:** modificatori presenti in modifiers.txt
- ❑ **Congiunzione “but”:** può invertire la polarità
- ❑ **Analisi 3-grammi:** permette di catturare le negazioni e invertire la polarità

3. DATASET



DATASET

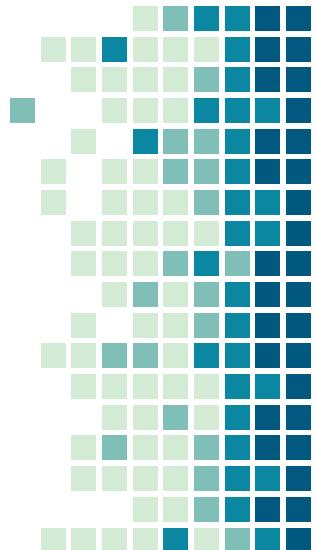
Recensioni di Amazon da *Julian McAuley, University of California San Diego.*

Il Dataset consiste in due file:

- raw review data (18gb)** - 59.12 million **reviews**
- metadata** (3.1gb) - metadata for 9.4 million **products**

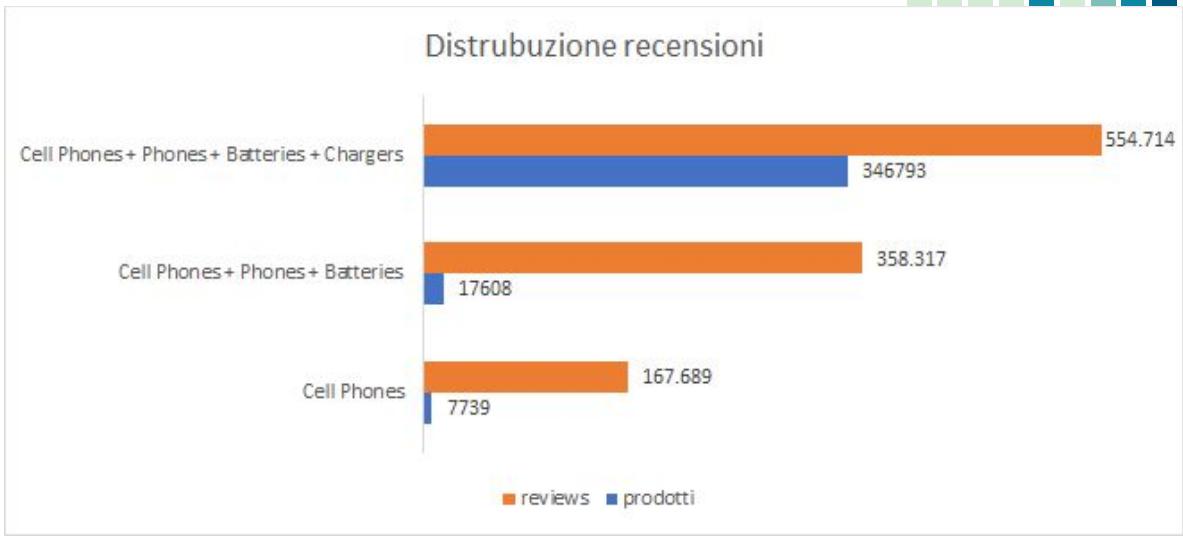
L'analisi della polarità si concentra sulla categoria di prodotti:
Cell-Phones & Accessories riducendo quindi il numero di prodotti a 350.312

DATASET



Abbiamo effettuato una serie di analisi per capire quale fosse il metodo migliore per selezionare un ulteriore sottoinsieme del dataset (per ovviare a problemi di risorse del DB centralizzato) evidenziando che :

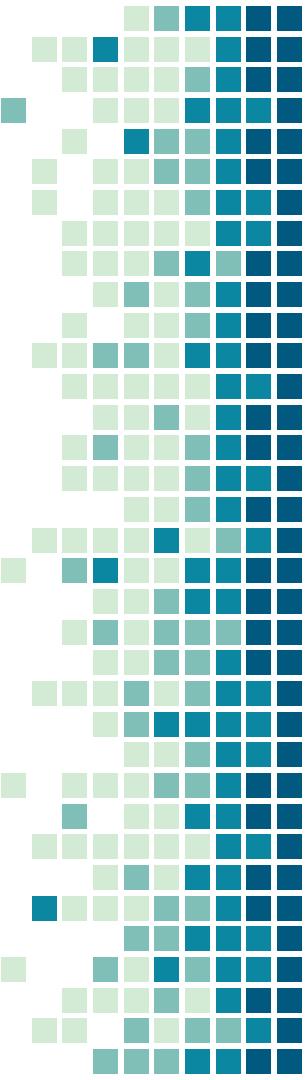
- ❑ Non conviene selezionare per brand poiché l'81% dei prodotti non ha un brand associato.
- ❑ Conviene utilizzare le sottocategorie (nel nostro caso *CellPhones, Phones, Battery, Chargers*)



4. DATABASES



DATABASES - MONGODB

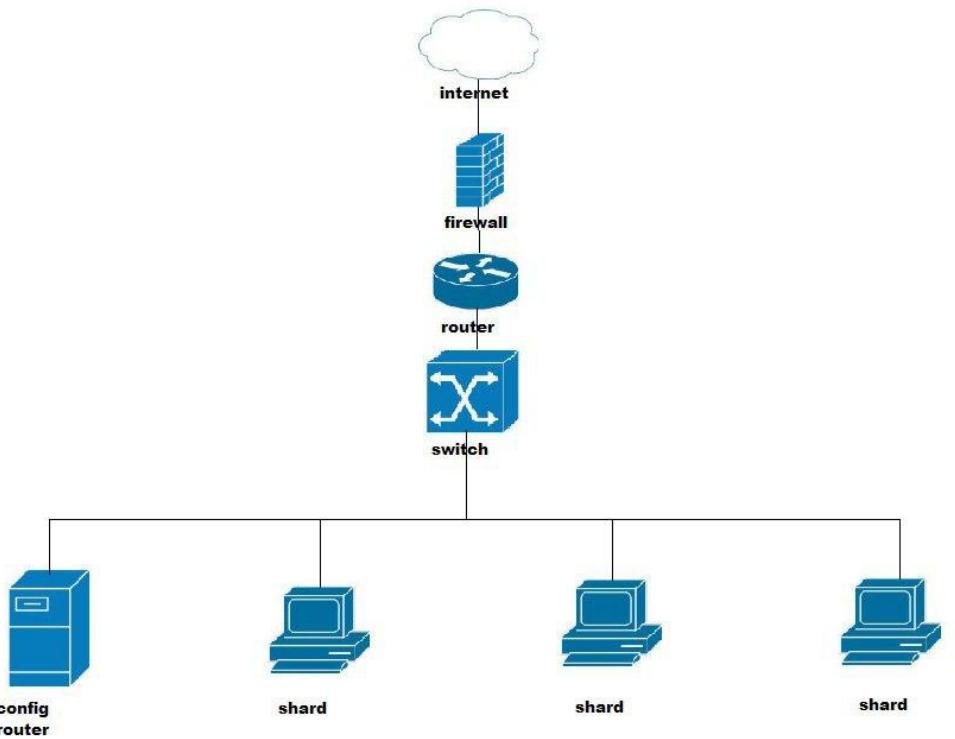


Cluster composto da 4 nodi:

- Master** : ruolo di **server config** e **router**
- Shard 1** : nodo sharding (contiene i dati)
- Shard 2** : nodo sharding (contiene i dati)
- Shard 3** : nodo sharding (contiene i dati)

Per semplicità e risorse di calcolo limitate abbiamo deciso di non replicare i dati, inoltre il *server config* è unico (*one-point of failure*) e risiede sulla stessa macchina del *router*.

DATABASES - MONGODB



DATABASES - POSTGRES

Database relazionale composto da 2 tavelli :

- ❑ **Products** : contiene i dati relativi ai prodotti presi in considerazione.
- ❑ **Reviews** : contiene i dati relativi alle recensioni dei prodotti della tabella products.

Inoltre abbiamo deciso di eseguire l'elaborazione sui DB Relazionali in 2 modi, usando :

- ❑ **Librerie JDBC** : implementando i metodi del DB con un approccio "classico" usando Java (server centralizzato)
- ❑ **Spark** : usando il framework spark per interrogare ed elaborare le tavelli del DB in modo partizionato e distribuito (su cluster)

5. SPARK



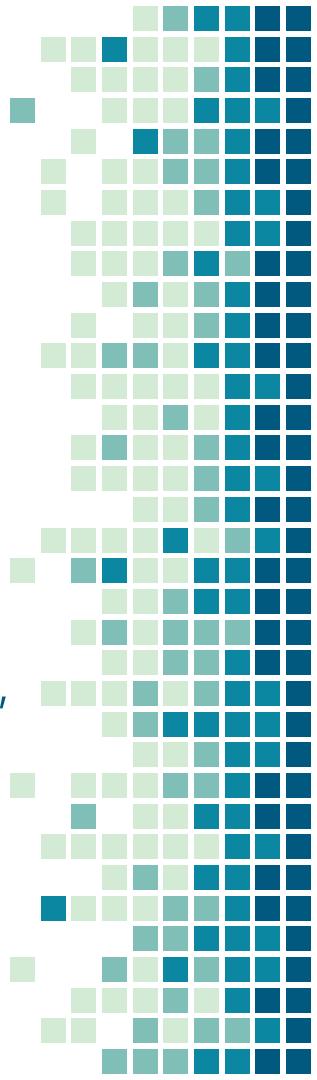
SPARK

Si tratta di un framework pienamente compatibile con MongoDB e Postgres.

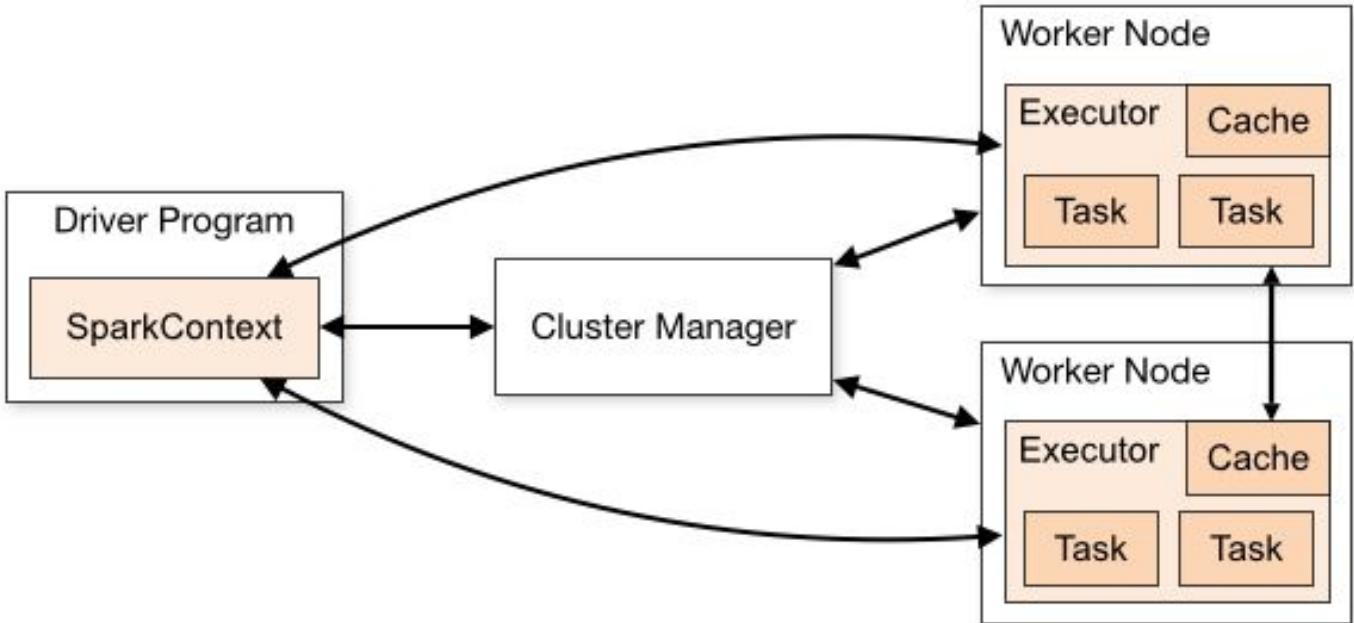
- ❑ Usato poiché le funzioni Map/Reduce di MongoDB sono implementate in Javascript, linguaggio non adatto ad applicazioni complesse.
- ❑ Permette di parallelizzare i calcoli anche con DB Relazionali.
- ❑ Fornisce numerose altre funzionalità (ES count, filter, groupBy)
- ❑ Gestisce il calcolo il più possibile in memoria centrale rivelandosi più efficiente di altri framework che utilizzano l'approccio Map/Reduce (prestazioni fino a 100 volte migliori per talune applicazioni)

SPARK

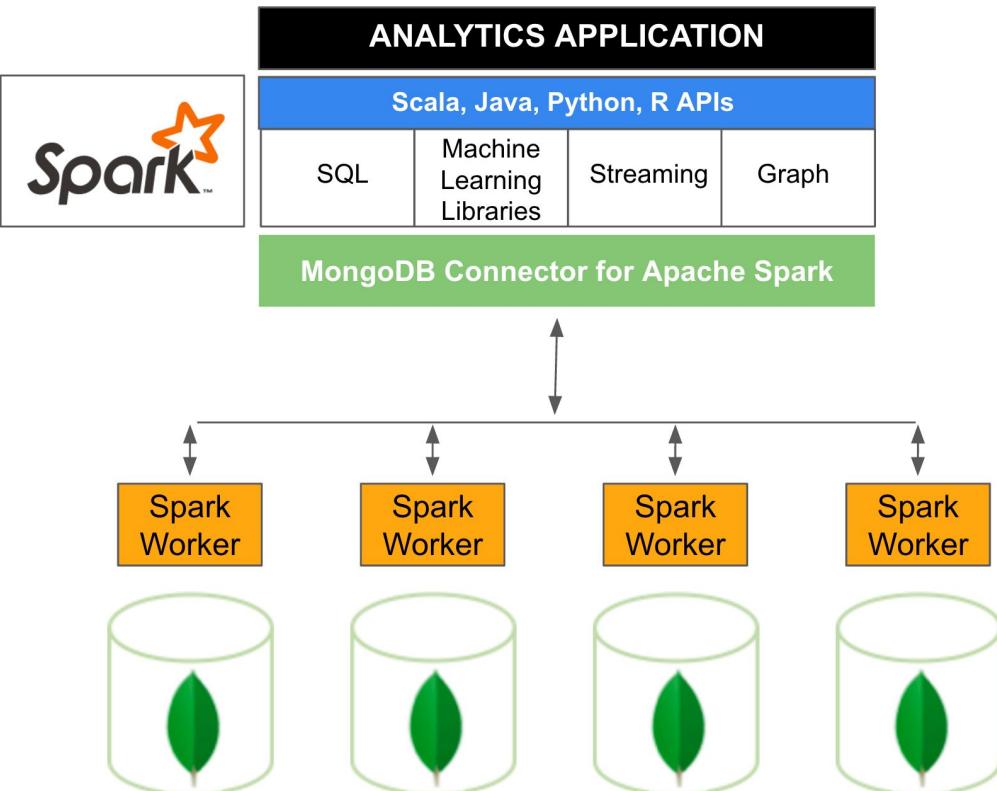
- ❑ RDD: unità base di lavoro, è una variabile distribuita di dati omogenei
- ❑ Trasformazioni: operazioni che applicano una funzione ad un RDD creandone uno nuovo (ES map, filter)
- ❑ Azioni: operazione che riduce un RDD ad una variabile Java, ossia non distribuita (ES reduce, count)
- ❑ Dataset e Dataframe: strutture dati pseudo-relazionali, distribuite, contenenti rispettivamente oggetti arbitrari e oggetti Row
- ❑ Valutazione lazy
- ❑ Possibile richiedere caching su disco, RAM o entrambi



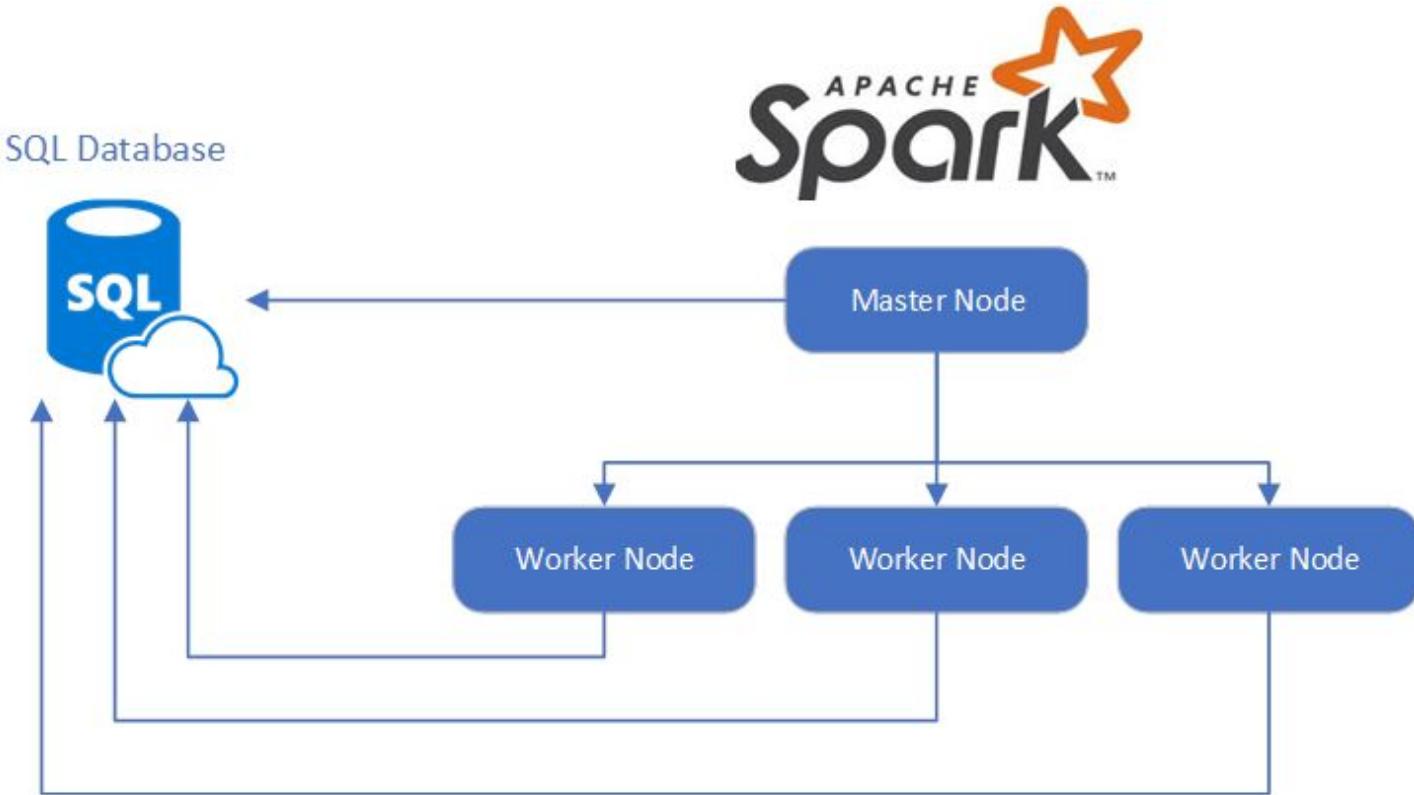
SPARK



INTEGRAZIONE SPARK-MONGODB



INTEGRAZIONE SPARK-POSTGRES

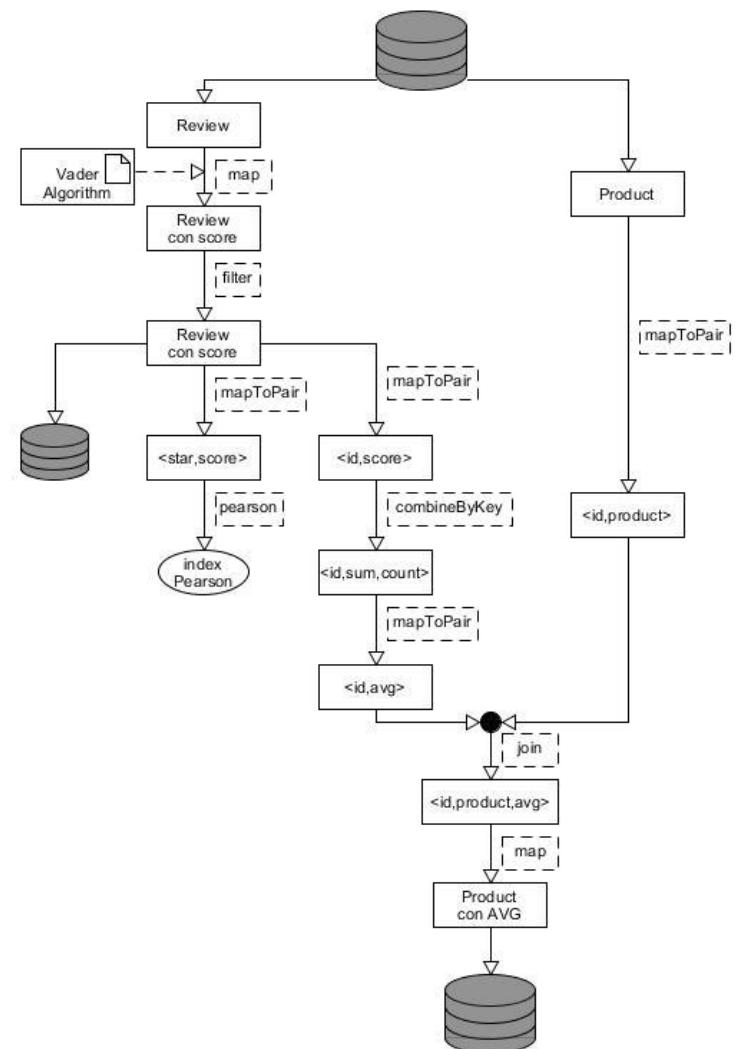
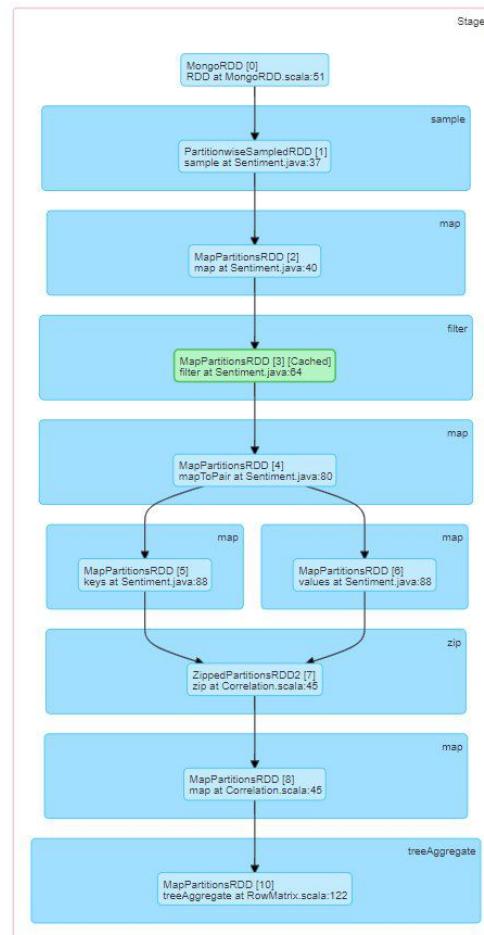


6. DIAGRAMMI DI FLUSSO



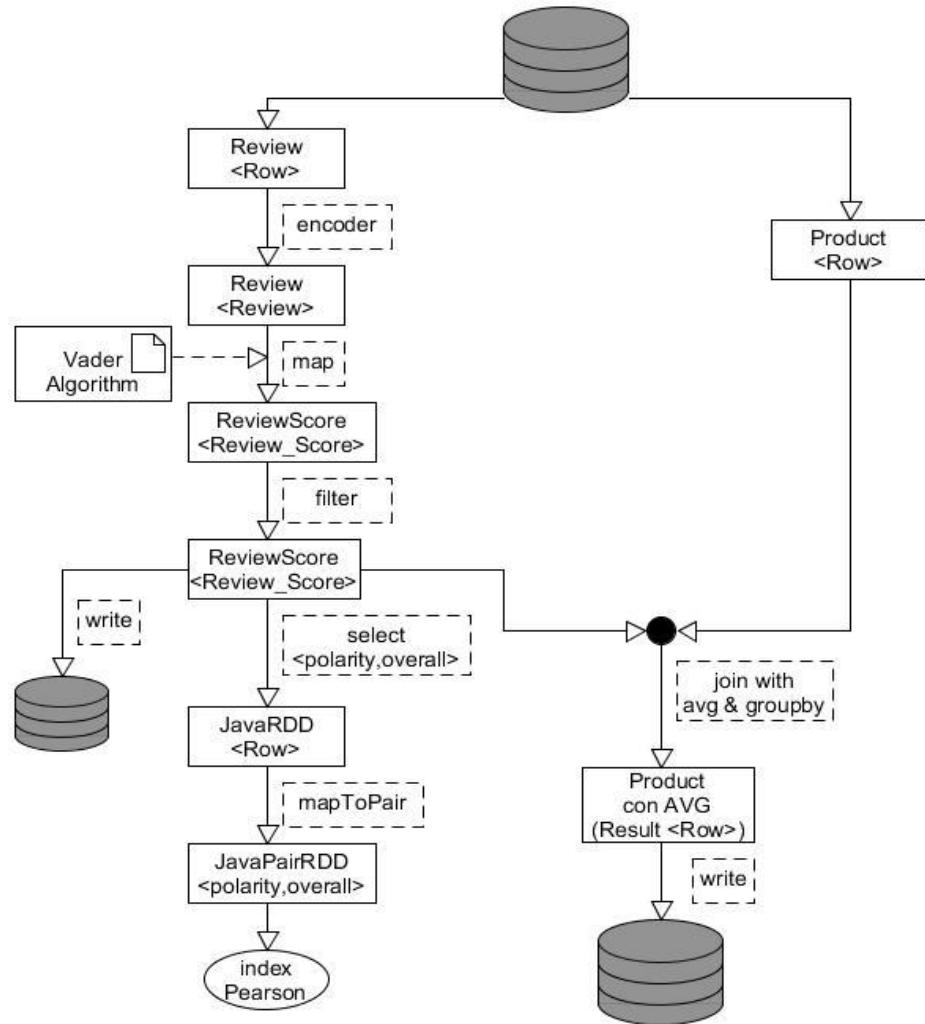
DIAGRAMMI DI FLUSSO NOSQL

Nella figura a sinistra è riportato il diagramma di flusso generato da spark (ogni nodo corrisponde ad un task), nella figura a destra invece è riportato il diagramma di flusso del codice



DIAGRAMMI DI FLUSSO POSTGRES-SPARK

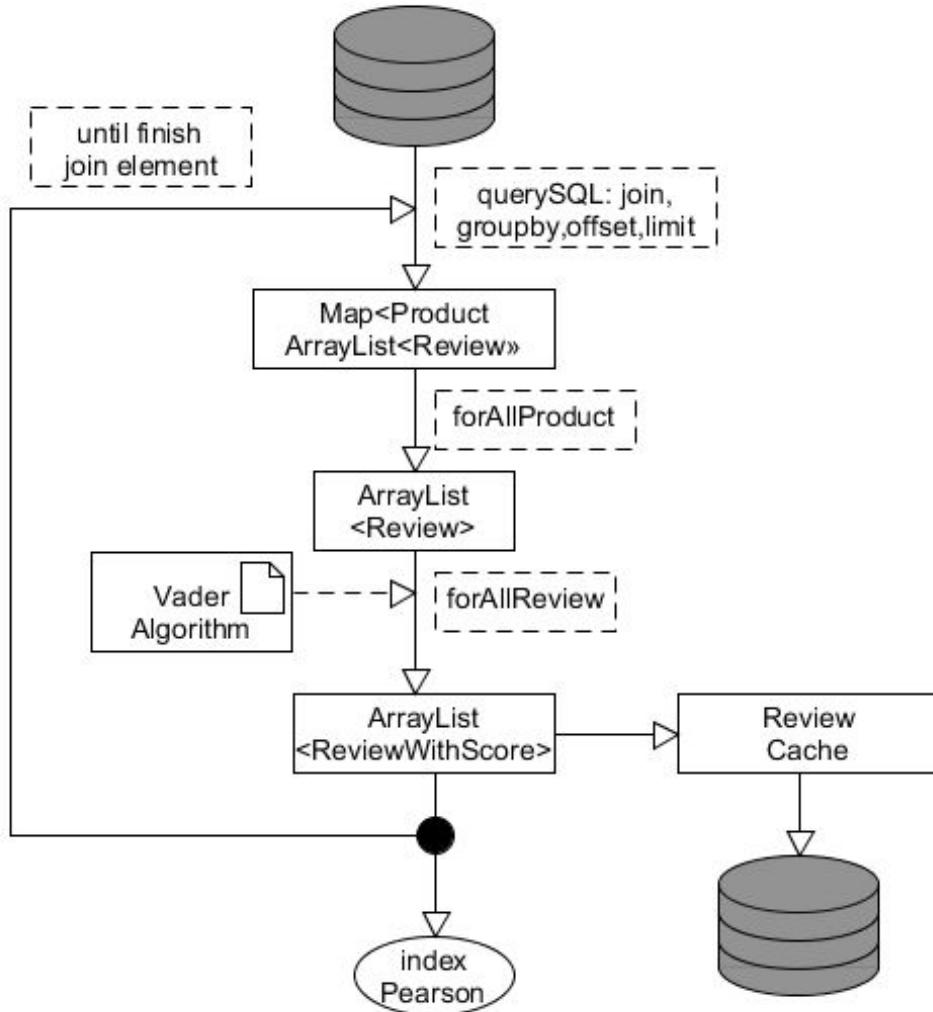
Diagramma di flusso del codice implementato su POSTGRES facendo uso della piattaforma SPARK



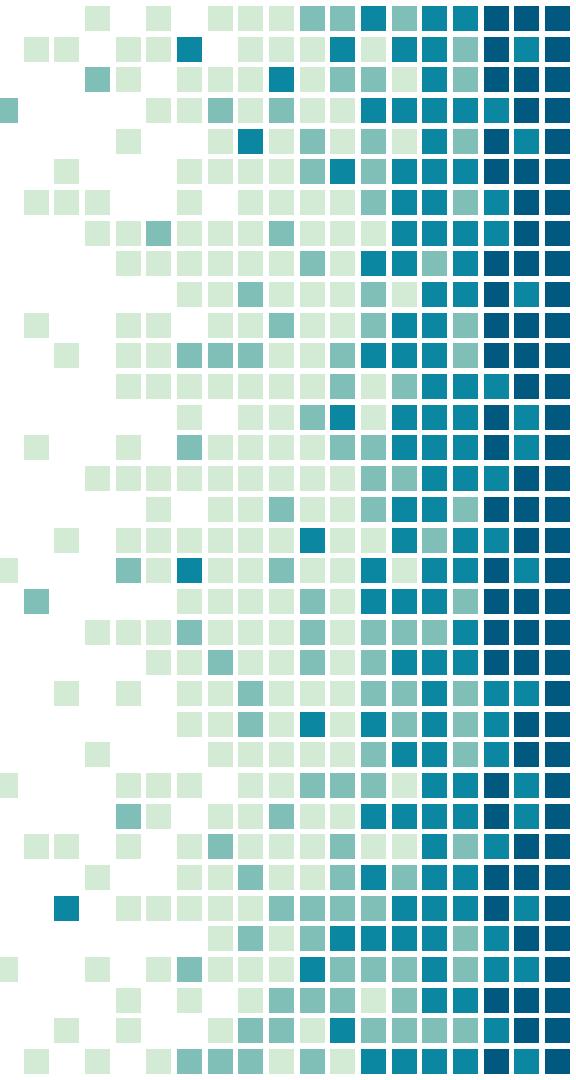
DIAGRAMMI DI FLUSSO POSTGRES-JDBC

Diagramma di flusso del codice implementato su POSTGRES facendo uso solamente della libreria JDBC

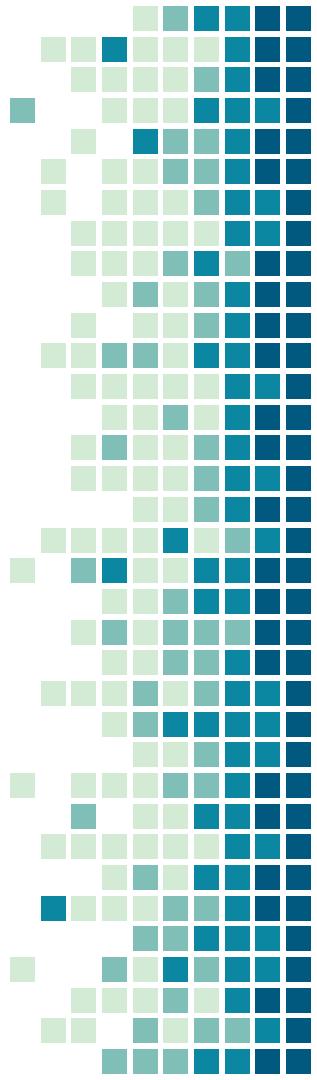
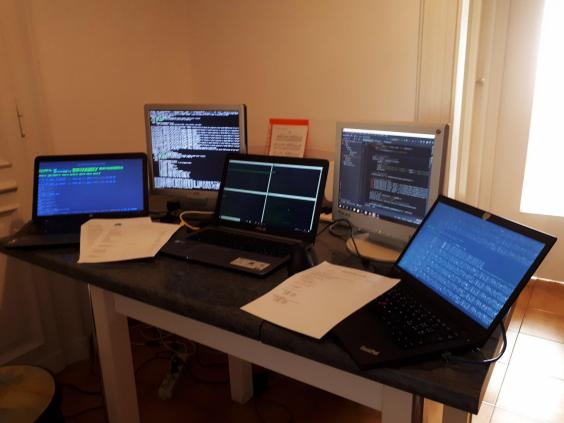
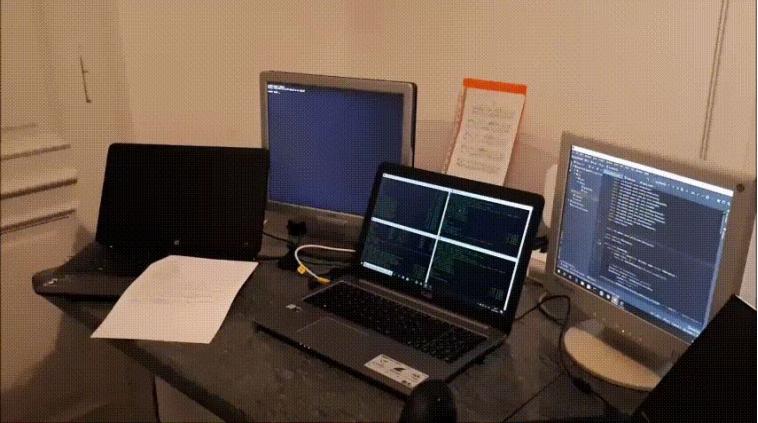
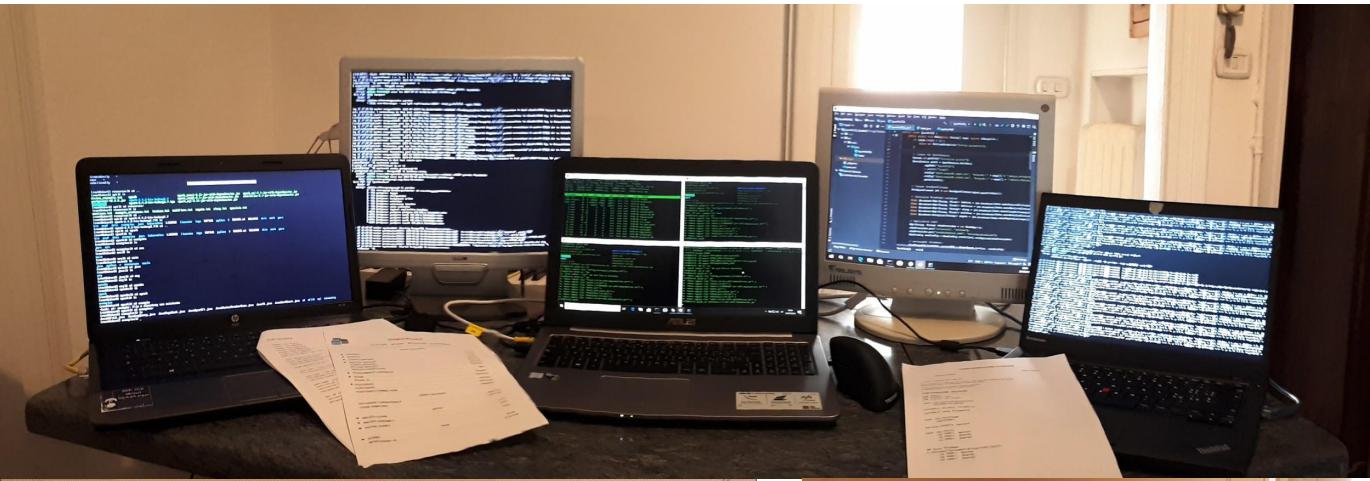
Grazie al sistema di caching implementato è stato possibile ridurre dell' 88% il tempo d'esecuzione



7. ESECUZIONE



ESECUZIONE - IL CLUSTER



ESECUZIONE - SPARK NOSQL

SPARK 2.4.2

Jobs Stages Storage Environment Executors SQL

SparkNoSQL application UI

User: rogyf
Total Uptime: 20 min
Scheduling Mode: FIFO
Completed Jobs: 5

Event Timeline

Enable zooming

Executors

- Added
- Removed

Jobs

- Succeeded
- Failed
- Running

11:42 11:43 11:44 11:45 11:46 11:47 11:48 11:49 11:50

Tue 23 July

foreachPartition at MongoSpark.scala:117 (Job 0)

Executor driver added

treeAgg

treeAgg

foreach

foreachPartition at MongoSpark.scala:117 (Job 0)

foreachPartition at MongoSpark.scala:117

treeAggregate at RowMatrix.scala:122

treeAggregate at RowMatrix.scala:122

treeAggregate at RowMatrix.scala:433

Submitted Duration Stages: Succeeded/Total Tasks (for all stages): Succeeded/Total

2019/07/23 11:50:07 17 s 4/4 25/25

2019/07/23 11:49:47 19 s 2/2 10/10

2019/07/23 11:49:30 17 s 2/2 10/10

Stage Id Description Submitted Duration Tasks: Succeeded/Total Input Output Shuffle Read Shuffle Write

9 foreachPartition at MongoSpark.scala:117 +details 2019/07/23 11:50:19 5 s 8/8 24.9 MB

Application ID Name Cores Memory per Executor Submitted Time User State Duration

app-20190722170712-0002 SparkNoSQL 11 1024.0 MB 2019/07/22 17:07:12 root FINISHED 4.5 min

ESECUZIONE - SPARK NOSQL

Apache Spark 2.4.2 Jobs Stages Storage Environment Executors SQL

Apache Spark 2.4.2 Jobs Stages Storage Environment Executors SQL

SparkNoSQL application UI

SparkNoSQL application UI

User Total Sched Comp Event Timeline DAG Visualization

Details for Job 4

Status: SUCCEEDED
Completed Stages: 4

- Event Timeline
- DAG Visualization

Stage 6

map

filter

map

Stage 7

combineByKeyWithClassTag

map

Stage 8

map

Stage 9

join

map

Completed Stages (4)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
9	foreachPartition at MongoSpark.scala:117	+details 2019/07/23 11:50:19	5 s	8/8		24.9 MB		
Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration	
app-20190722170712-0002	SparkNoSQL	11	1024.0 MB	2019/07/22 17:07:12	root	FINISHED	4.5 min	

ESECUZIONE - SPARK NOSQL

The screenshot shows the Apache Spark 2.4.2 UI interface. The top navigation bar includes tabs for Jobs, Stages, Storage, Environment, Executors, and SQL. A secondary tab bar for the "SparkNoSQL application UI" is also present. The main content area displays the "Spark Master at spark://master:7077" details. It shows the following information:

- Status: **Running**
- URL: `spark://master:7077`
- Alive Workers: 3
- Cores in use: 12 Total, 0 Used
- Memory in use: 13.9 GB Total, 0.0 B Used
- Applications: 0 Running, 3 Completed
- Drivers: 0 Running, 3 Completed
- Status: ALIVE

Below this, there are sections for **Workers (3)**, **Running Applications (0)**, **Running Drivers (0)**, and **Completed Applications (3)**. Each section contains a table with specific details.

Worker Id	Address	State	Cores	Memory
worker-20190722163018-192.168.1.7-35284	192.168.1.7:35284	ALIVE	4 (0 Used)	6.3 GB (0.0 B Used)
worker-20190722163024-192.168.1.6-45381	192.168.1.6:45381	ALIVE	4 (0 Used)	1024.0 MB (0.0 B Used)
worker-20190722163026-192.168.1.8-36409	192.168.1.8:36409	ALIVE	4 (0 Used)	6.5 GB (0.0 B Used)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
No applications running.							

Submission ID	Submitted Time	Worker	State	Cores	Memory	Main Class
No drivers running.						

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20190722170712-0002	SparkNoSQL	11	1024.0 MB	2019/07/22 17:07:12	root	FINISHED	4.5 min

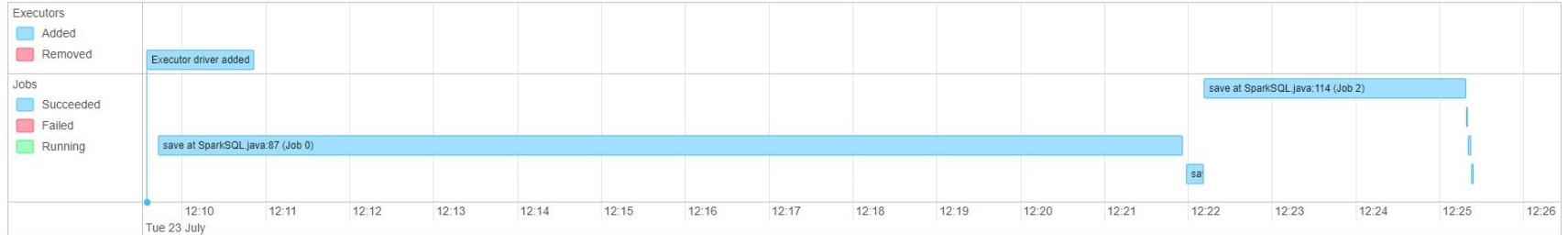
ESECUZIONE - SPARK SQL

Spark Jobs (?)

User: rogyf
Total Uptime: 16 min
Scheduling Mode: FIFO
Completed Jobs: 6

▼ Event Timeline

Enable zooming



Completed Jobs (6)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
5	treeAggregate at RowMatrix.scala:122 treeAggregate at RowMatrix.scala:122	2019/07/23 12:25:22	2 s	1/1	1/1
4	treeAggregate at RowMatrix.scala:433 treeAggregate at RowMatrix.scala:433	2019/07/23 12:25:20	2 s	1/1	1/1
3	first at RowMatrix.scala:61 first at RowMatrix.scala:61	2019/07/23 12:25:18	1 s	1/1	1/1
2	save at SparkSQL.java:114 save at SparkSQL.java:114	2019/07/23 12:22:11	3.1 min	2/2 (2 skipped)	400/400 (2 skipped)
1	save at SparkSQL.java:114 save at SparkSQL.java:114	2019/07/23 12:21:58	13 s	3/3	202/202
0	save at SparkSQL.java:87 save at SparkSQL.java:87	2019/07/23 12:09:42	12 min	1/1	1/1

> Tablespaces

> VM

> centOS

app-20190722172024-0003

SparkSQL

11

1024.0 MB

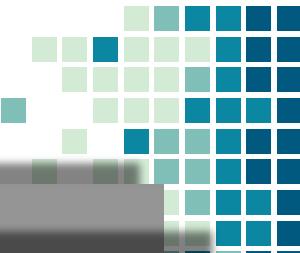
2019/07/22 17:20:24

TOOL

FINISHED

17 min

ESECUZIONE - SPARK SQL



Spark Jobs (?)

User: root
Total: 0 Schemas: 0

pgAdmin 4

File ▾ Object ▾ Tools ▾ Help ▾

Browser

- Extensions
- Foreign Data Wrappers
- Languages
- Schemas (1)
 - public
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Procedures
 - Sequences
 - Tables (2)
 - products
 - reviews
 - Trigger Functions
 - Types
 - Views
- postgres
- raadb
- Login/Group Roles
- Tablespaces
- VM
- centOS

Dashboard Properties SQL Statistics Dependencies Dependents

Database sessions

Transactions per second

Tuples in

Tuples out

Block I/O

Server activity

Sessions Locks Prepared Transactions

	PID	User	Application	Client	Backend start	State	Wait event	Blocking PIDs
✖	956	postgres	pgAdmin 4 - DB:maadb	::1	2019-07-23 11:13:06 CEST	active		

31

app-20190722172524-0005

SparkSQL

11 1024.0 MB

2019/07/22 17:25:24

root FINISHED 17 min

ESECUZIONE - SPARK SQL

Spark Jobs (?)

User: root
Total: 1 pgAdmin 4 File Object Tools Help

Com Browser

Apache Spark 2.4.2

Spark Master at spark://master:7077

URL: spark://master:7077
Alive Workers: 3
Cores in use: 12 Total, 0 Used
Memory in use: 13.9 GB Total, 0.0 B Used
Applications: 0 Running, 4 Completed
Drivers: 0 Running, 4 Completed
Status: ALIVE

▼ Workers (3)

Worker Id	Address	State	Cores	Memory
worker-20190722163018-192.168.1.7-35284	192.168.1.7:35284	ALIVE	4 (0 Used)	6.3 GB (0.0 B Used)
worker-20190722163024-192.168.1.6-45381	192.168.1.6:45381	ALIVE	4 (0 Used)	1024.0 MB (0.0 B Used)
worker-20190722163026-192.168.1.8-36409	192.168.1.8:36409	ALIVE	4 (0 Used)	6.5 GB (0.0 B Used)

▼ Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	----------------	------	-------	----------

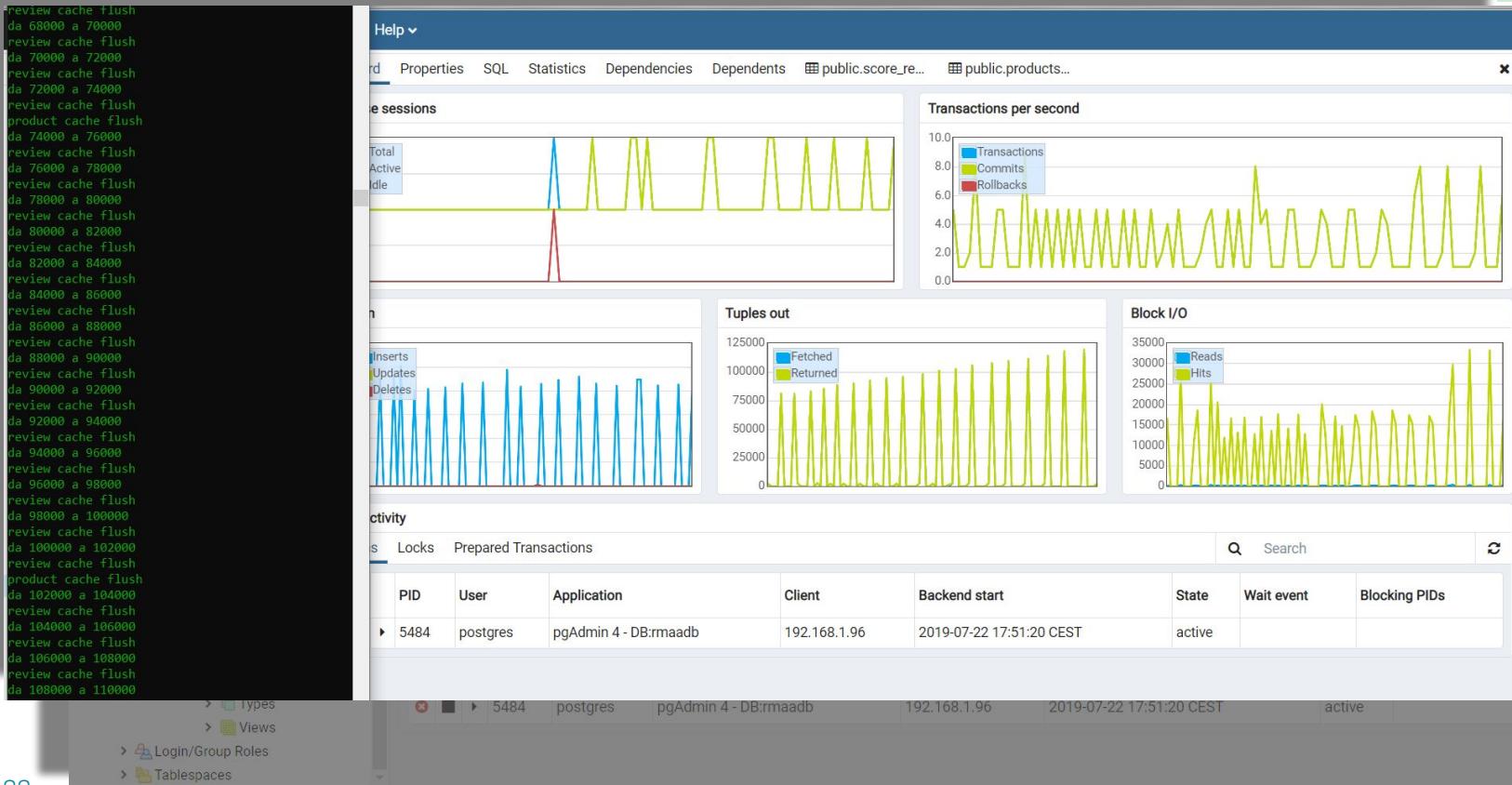
▼ Running Drivers (0)

Submission ID	Submitted Time	Worker	State	Cores	Memory	Main Class
---------------	----------------	--------	-------	-------	--------	------------

▼ Completed Applications (4)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20190722172524-0003	SparkSQL	11	1024.0 MB	2019/07/22 17:25:24	root	FINISHED	17 min

ESECUZIONE - JDBC ONLY



ESECUZIONE - JDBC ONLY

review cache flush
da 68000 a 70000
review cache flush
da 70000 a 72000
review cache flush
da 72000 a 74000
review cache flush
da 74000 a 76000
review cache flush
da 76000 a 78000
review cache flush
da 78000 a 80000
review cache flush
da 80000 a 82000
review cache flush
da 82000 a 84000
review cache flush
da 84000 a 86000
review cache flush
da 86000 a 88000
review cache flush
da 88000 a 90000
review cache flush
da 90000 a 92000
review cache flush
da 92000 a 94000
review cache flush
da 94000 a 96000
review cache flush
da 96000 a 98000
review cache flush
da 98000 a 100000
review cache flush
da 100000 a 102000
review cache flush
da 102000 a 104000
review cache flush
da 104000 a 106000
review cache flush
da 106000 a 108000

pgAdmin 4

File ▾ Object ▾ Tools ▾ Help ▾

Browser

- Servermaadb
 - Casts
 - Catalogs
 - Event Triggers
 - Extensions
 - Foreign Data Wrappers
 - Languages
 - Schemas (1)
 - public
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Procedures
 - Sequences
 - Tables (2)
 - products
 - reviews
 - Trigger Functions
 - Types
 - Views
 - Login/Group Roles
 - Tablespaces

Dashboard Properties SQL Statistics Dependencies Dependents public.score_re... public.products...

Database sessions

Total, Active, Idle

Transactions per second

Transactions

Tuples in

Inserts

Tuples out

Fetched, Returned

Block I/O

Reads, Hits

Server activity

	Sessions	Locks	Prepared Transactions					
	PID	User	Application	Client	Backend start	State	Wait event	Blocking PIDs
⌚	5484	postgres	pgAdmin 4 - DB:rmaadb	192.168.1.96	2019-07-22 17:51:20 CEST	active		

Search

7. RISULTATI



RISULTATI

I risultati che seguono si riferiscono alle seguenti configurazioni:

- Server centralizzato (Windows)
 - Intel Core i7 3.1 GHz RAM 12 GB
- Cluster di 4 nodi (CentOS)
 - Intel (VM) 3.1 GHz RAM 2 GB (worker - shard)
 - Intel Core i5 2.4 GHz RAM 8 GB (worker - shard)
 - Pentium Dual 1.6 GHz RAM 2 GB (master)
 - Intel Core i3 2.4 GHz RAM 8 GB (worker - shard)

Tot 12 core, 13.9 GB memoria, 3 worker, 3 shard

RISULTATI - LOCALE

L'esecuzione locale simula un cluster utilizzando diversi thread e processi, si tenga presente che il processore ha 2 core con 4 thread ciascuno (possibile simulare 8 core)

Soluzione	Tempo esecuzione
Spark-MongoDB	8 min
Spark-PostgreSQL	16 min
JDBC only	9 min

RISULTATI - CLUSTER

Esecuzione su cluster utilizza 11 core e 1024 MB per Executors
(esecutori del codice nei nodi del cluster)

Soluzione	Tempo esecuzione
Spark-MongoDB	4 min
Spark-PostgreSQL	17 min
JDBC only	29 min

RISULTATI - PEARSON

I coefficiente di correlazione Pearson tra le stelle attribuite dall'utente (considerato come gold-value) nella recensione e la polarità calcolata mediante l'algoritmo VADER è pari a 0.65

Il risultato si potrebbe migliorare integrando risorse esterne (ad esempio *WordNet*, *BabelNet*, *ConceptNet*) per comprendere ironia e sarcasmo

CREDITS

Dataset:

R. He, J. McAuley. "Modeling the visual evolution of fashion trends with one-class collaborative filtering". WWW, 2016

J. McAuley, C. Targett, J. Shi, A. van den Hengel. "Image-based recommendations on styles and substitutes". SIGIR, 2015

Vader:

Hutto, Clayton J. and Eric Gilbert. "VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text." ICWSM, 2014

THANKS!