Università Telematica Internazionale
Uninettuno

INGEGNERIA INFORMATICA
Corsi di Laurea Magistrale - Big Data

Introduzione ai Big Data

Progetto su R, MapReduce e Hadoop

"Contabilità ditta"

Docente:
Prof. Pirrone Daniele

Studente:
Emanuele Coltro
mat: 671HHHINGINFOR

Anno Accademico 2022-2023

Abstract

Il presente paper ha l'obiettivo di esaminare le potenzialità e le performance degli strumenti utilizzati per la gestione e l'analisi dei Big Data. Viene approfondito il processo che porta alla produzione di report grafici, tabelle e schemi a partire dai dati di un problema o di una esigenza. La relazione fornisce informazioni dettagliate su come impostare un progetto, dalla scelta e impostazione dell'ambiente di sviluppo alla contestualizzazione delle decisioni prese. Inoltre, vengono allegati i codici per la generazione dei risultati e dei grafici.

Il progetto in esame mira a soddisfare l'esigenza dell'azienda (fittizzia) STECCAPARAPETUTTI S.R.L. di risolvere un problema di contabilità mediante l'estrazione di alcune feature dai dati. A tal fine, sono stati impiegati strumenti avanzati per l'elaborazione dei dati, tra cui algoritmi di MapReduce e tecniche di visualizzazione avanzata.

L'analisi dei dati è stata eseguita utilizzando il linguaggio R per una prima analisi, seguito dall'utilizzo di un tool scritto in JavaScript per prototipare il sistema di MapReduce e risolvere il problema.

Indice

1 Introduzione

In questa fase, forniremo le basi dei concetti teorici e degli strumenti utilizzati nel corso del progetto. Vedremo in dettaglio come sia stato impostato l'ambiente di sviluppo e come sia stato affrontato il problema in analisi.

1.1 Richieste

La società fittizia STECCAPARAPETUTTI S.R.L. richiede l'analisi dei dati delle vendite del sistema di contabilità dell'anno precedente. I dati sono contenuti in un file CSV strutturato in modo omogeneo. L'analisi richiede il calcolo della media e della varianza delle vendite per ogni mese di ogni anno, l'identificazione del mese di ogni anno con la maggiore e minore vendita. Per ogni lavoro, è necessaria la documentazione dell'implementazione in R e dell'implementazione del paradigma MapReduce di Hadoop. Il risultato finale deve essere presentato sotto forma di un rapporto completo.

1.2 Introduzione all'ambiente

1.2.1 MapReduce

L'algoritmo MapReduce è un modello di programmazione e un'infrastruttura di elaborazione distribuita utilizzata per elaborare e generare informazioni da enormi quantità di dati in modo efficiente, sicuro e scalabile. È stato introdotto da Google[1] e ha giocato un ruolo fondamentale nel campo del data processing su larga scala.

L'idea alla base di MapReduce è suddividere un grande task di elaborazione dei dati in due fasi principali: la fase di "map" e la fase di "reduce" (più una fase intermedia totalmente automatizzata):

- **Map:** Durante la fase di MapReduce, i dati di input vengono suddivisi in piccoli frammenti (split) e passati a un insieme di processi paralleli chiamati "Worker". Ognuno di essi esegue una funzione di mapping definita dall'utente che prende in ingresso un dato e produce una serie di coppie chiave-valore intermedie. Queste coppie rappresentano il risultato dell'elaborazione dei dati nella fase di mapping.

- **Shuffle e Sort:** Dopo la fase di map, il sistema raggruppa le coppie chiave-valore in base alle chiavi e le ordina nella fase di Shuffle e Sort. Questo è un passaggio cruciale poiché consente ai dati correlati di essere inviati allo stesso processo di "reduce".

- **Reduce:** Nell'ultima fase il sistema assegna le coppie chiave-valore raggruppate nuovamente ai processi worker. Ogni processo esegue una funzione di "reduce" definita dall'utente che agisce sulle coppie chiave-valore correlate e produce i risultati finali dell'elaborazione.

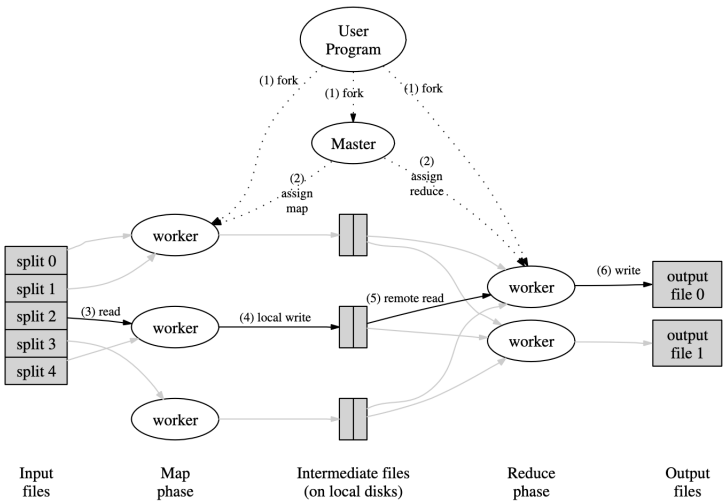

Figura 1: Schema del processo di MapReduce tratto dal paper di Google

L'obiettivo di MapReduce è la parallelizzazione e la distribuzione del carico di lavoro su un cluster di computer, consentendo di elaborare grandi quantità di dati in modo più veloce ed efficiente rispetto a un approccio sequenziale tradizionale. Questo modello di programmazione nasconde molti dettagli complessi dell'elaborazione distribuita, semplificando la creazione di applicazioni che possono sfruttare l'elaborazione su larga scala senza doversi preoccupare delle questioni di basso livello.

Questo ha portato negli anni alla nascita di tecnologie e framework, come Apache Hadoop e Apache Spark, che forniscono funzionalità di MapReduce ma più avanzate e versatili per l'elaborazione distribuita dei dati.

1.2.2 Hadoop

Apache Hadoop è un framework open-source sviluppato per consentire l'elaborazione distribuita di grandi quantità di dati su cluster di computer. È basato sulla paper originale di Google su MapReduce e il file system distribuito Google File System (GFS). Hadoop è progettato per gestire l'elaborazione di dati in parallelo su macchine commodity (hardware relativamente economico e accessibile).

Il cuore di Hadoop[3] è composto da due componenti principali: Hadoop Distributed File System (HDFS) e il framework di elaborazione MapReduce.

1. Hadoop Distributed File System (HDFS): HDFS è il sistema di archiviazione distribuito di Hadoop. Si basa su un modello di architettura master-slave in cui un nodo master, chiamato "NameNode", gestisce i metadati del file system e tiene traccia di dove sono archiviati i dati. I nodi slave, chiamati "DataNode", contengono i blocchi di dati reali. I dati vengono suddivisi in blocchi e replicati su vari DataNode per garantire la disponibilità e l'affidabilità.

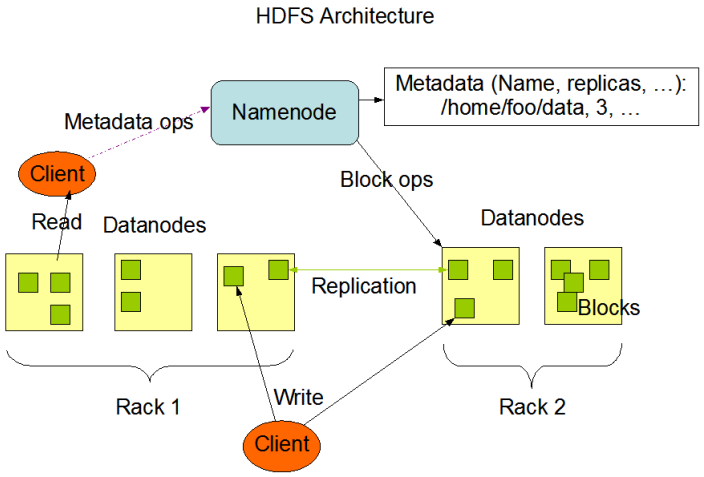

Figura 2: Schema dell'architettura HDFS tratto dal sito di Hadoop

2. MapReduce: Come descritto in precedenza, MapReduce è un modello di programmazione per l'elaborazione distribuita. Hadoop implementa questo modello consentendo agli sviluppatori di scrivere programmi MapReduce per l'elaborazione dei dati su cluster. Il framework si occupa della distribuzione delle attività di map e reduce su diversi nodi del cluster, dell'ordinamento e dell'aggregazione dei risultati intermedi e della gestione dei fallimenti dei nodi.

I punti di forza di Hadoop includono la scalabilità orizzontale, la tolleranza ai guasti, l'economia hardware, l'ecosistema di strumenti, l'elaborazione su larga scala, la flessibilità nei tipi di dati e l'elaborazione di dati grezzi. Tuttavia, Hadoop ha anche alcuni svantaggi come la complessità nella configurazione e gestione di un cluster, la latenza, la memoria limitata, l'approccio orientato ai batch, la complessità della programmazione e la concorrenza da parte di framework alternativi come Apache Spark.

1.2.3 R e RStudio

R è un linguaggio di programmazione e un ambiente di sviluppo utilizzati principalmente per l'analisi statistica e la visualizzazione dei dati, progettato per lavorare con dataset di varie dimensioni e complessità. Offre un'ampia gamma di funzioni statistiche, algoritmi e pacchetti per l'analisi dei dati, il machine learning, l'analisi delle serie temporali e altro ancora. Inoltre, R fornisce potenti strumenti di visualizzazione che consentono di creare grafici e grafici per rappresentare i dati in modo efficace.

RStudio è un ambiente di sviluppo integrato (IDE) progettato specificamente per lavorare con il linguaggio di programmazione R. Come R[5] anche RStudio[4] è un progetto open source che ha lo scopo di offrire un'interfaccia utente intuitiva e ben organizzata che semplifica la scrittura del codice R, l'analisi dei dati e la creazione di visualizzazioni.

1.2.4 Python

Python[2] è un linguaggio di programmazione che ha catturato l'attenzione di sviluppatori di tutto il mondo grazie alla sua natura versatile e alla sua sintassi chiara e leggibile. Creato da Guido van Rossum[6] e presentato nel lontano 1991, Python ha continuato a guadagnare popolarità nel corso degli anni, diventando uno dei linguaggi più utilizzati e apprezzati nella comunità dello sviluppo software.

Ciò che distingue Python è il suo approccio alla scrittura del codice. La sua sintassi è strutturata in modo simile al linguaggio naturale, rendendo il codice scritto in Python quasi come una conversazione tra lo sviluppatore e la macchina. Questa semplicità e leggibilità non solo agevolano la creazione del codice, ma anche la comprensione dello stesso da parte di altri sviluppatori, favorendo una collaborazione più agevole.

Python (come anche R) è un linguaggio interpretato, il che significa che non richiede una fase di compilazione separata. Questo aspetto favorisce un approccio di sviluppo più rapido e interattivo, consentendo agli sviluppatori di scrivere e testare il codice in modo immediato, senza dover attendere processi di compilazione lungo.

1.2.5 Tool Javascript-html per MapReduce

Per evitare possibili complicazioni dovute alla configurazione di Hadoop, il Professore Daniele Pirrone ha creato uno script web chiamato "Tool MapReduce". Questo strumento consente di simulare un programma MapReduce attraverso un browser generico su Internet.

Lo strumento è sviluppato in Javascript con interfaccia HTML, rendendolo compatibile con i principali sistemi operativi e browser. Il codice è rilasciato sotto licenza GPL.

Lo strumento è semplicemente uno strumento didattico di supporto per gli studenti e quindi:

- Consente l'utilizzo di file di input che sono limitati in dimensione alla capacità del computer su cui viene eseguito.
- Accetta solo file di testo come input (la maggior parte dei quali hanno un'estensione .txt).
- Simula il paradigma di programmazione parallela MapReduce, ma in realtà funziona in modalità stand-alone, non distribuita.

1.3 Preparazione dell'ambiente

Affrontiamo ora il processo di installazione e configurazione dell'ambiente di sviluppo su macchina virtuale, che si è rivelato necessario per una corretta comunicazione tra Hadoop, RStudio e Python.

Le principali motivazioni di questa scelta risiedono nel fatto che i packages di RHadoop per una corretta comunicazione tra RStudio e Hadoop non sono aggiornati alle ultime versioni [1].

La scelta di utilizzare la versione di Hadoop all'interno di una macchina virtuale anziché installarla direttamente sul sistema locale è stata influenzata da problematiche di gestione del DFS estendendo tutto il disco a HDFS andando quindi a creare conflitti.

Sfruttando il contesto isolato che offre la virtualizzazione si ha potuti sperimentare con un approccio trial and error diverse configurazioni che permettessero il corretto funzionamento di tutte le componenti. Sono state così fatte scelte specifiche per quanto riguarda le versioni dle sistema operativo e dei software da utilizzare, tenendo conto della compatibilità tra di essi e delle raccomandazioni della comunità.

Si è scelto Virtualbox come sistema di virtualizzazione in quanto software open source di ampia distribuzione.

1.3.1 Preparazione dell'ambiente virtuale

Il primo passo è stato creare una nuova macchina virtuale su VirtualBox dal sistema host assegnando risorse hardware adeguate.

[1] l'ultima release di RHadoop risale al 2015

Ruolo	Sistema Operativo	CPU	RAM	Spazio su disco
Host	macOS Ventura 13.4	2.3 GHz Intel Core i5 dual-core	16 GB 2133 MHz LPDDR3	500 GB
VM	Ubuntu 16.04	1 CPU	4 GB	80 GB

Tabella 1: Specifiche di sistema

È stata scelta una distribuzione Linux come sistema operativo ospite per sfruttare la flessibilità e le prestazioni offerte da questa piattaforma open source. Essendo infatti Apache Hadoop un sistema utilizzato principalmente lato server, la sua implementazione più diffusa e stabile è in ambiente UNIX.

1.3.2 Installazione di Hadoop

Per garanzie di compatibilità si è scelto la versione 2.6.5 di Apache Hadoop, andandolo a configurare in modalità Single Node Cluster. Successivamente si è anche cercato di aggiungere nodi al cluster ma con risultati poco stabili.[2]

1.3.3 Integrazione di RStudio

L'integrazione di RStudio è stata relativamente più semplice utilizzando la versione 1.0.153 del 2017 e versione di R 3.2.3 che ha permesso una installazione delle librerie di RHadoop senza conflitti.

1.3.4 Configurazione di Python

La configurazione di Python presente già nel sistema operativo offre le versioni 2.7 e 3.5.2 con i relativi comandi python e python3. L'unica configurazione necessaria è stata l'intallazione del gestore di pacchetti Python "pip" per l'installazione delle librerie utilizzate poi negli script.

1.3.5 Conclusione configurazione

L'installazione e la configurazione dell'ambiente di sviluppo su VirtualBox con Hadoop, RStudio e Python hanno richiesto un approccio meticoloso e la riso-

[2]In una configurazione iniziale si era utilizzata sia una soluzione "fisica" con due Raspberry Pi 3 Model B come nodi dati, sia macchine virtuali Ubuntu Server 22.04. Entrambe le soluzioni hanno presentato problematiche, quindi si è preferito tornare alla configurazione Single Node.

luzione di diverse sfide tecniche. Le scelte fatte durante il processo sono state guidate dalla necessità di garantire l'integrazione corretta dei componenti e l'ottimizzazione delle risorse disponibili.

2 Svolgimento

2.1 Analisi del problema

Abbiamo iniziato analizzando le richieste del problema da un punto di vista matematico. Per risolverle, bisogna essere in grado di trovare i documenti che contribuiscono all'incremento del fatturato e discriminare gli altri. In un secondo momento, è necessario calcolare la media (Eq.1) e la varianza (Eq.2) utilizzando le formule sottostanti.

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{1}$$

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{2}$$

Successivamente, sarà necessario calcolare l'ammontare del fatturato diviso per ogni mese di ogni anno. Una volta elaborate queste informazioni, si può procedere ad identificare il mese di ogni anno che ha il valore più alto e più basso, analizzandoli tramite le funzioni di minimo e massimo e raggruppando per anno.

2.2 Analisi del Dataset

Procediamo con l'analisi del dataset con i comandi

```
1    summary(df)
2    table(df$tipo)
```

che vengono utilizzati per valutare la tipologia dei dati e per estrarre i valori della prima colonna riferiti al tipo di documento. In prima analisi, ci rendiamo

Tipo di documento	Quantità
BUONO.PRELIEVO	2146
DDT	15
FATTURA	556
INVENTARIO	1
NOTA.DI.CREDITO	9
OFFERTA	622
PREVENTIVO	240
RICEVUTA	11

Tabella 2: Analisi delle tipologie di documento

conto che i documenti che apportano incremento al fatturato sono della tipologia

fattura e ricevuta. Andiamo quindi ad impostare un filtro che ci aiuta ad estrarre dal dataset i documenti interessati.

2.3 Risoluzione tramite codice R

A questo punto, procediamo con il calcolo della media (Allegato contabilita.r) utilizzando la funzione `aggregate`, aggregando in base ai mesi di ogni anno e applicando la funzione di calcolo della media (`FUN = mean`). Vengono fatti in seguito delle trasformazioni per estrarre dalla data le informazioni di anno e mese e renderle più facilmente leggibili. Successivamente, andiamo a calcolare la varianza utilizzando la stessa funzione `aggregate` e impostando `FUN = var`.

vendite_medie	data_formattata
179.93081	2016-Gen
131.92288	2016-Feb
222.55800	2016-Mar
182.44100	2016-Apr
200.41270	2016-Mag
143.58000	2016-Giu
185.62951	2016-Lug
66.66857	2016-Ago
119.33484	2016-Set
174.44197	2016-Ott

Figura 3: Estratto del dataset con la media

data_formattata	varianza
2016-Gen	110972.691
2016-Feb	38351.044
2016-Mar	295872.781
2016-Apr	191570.685
2016-Mag	196337.279
2016-Giu	69653.870
2016-Lug	160827.843
2016-Ago	5731.737
2016-Set	35471.358
2016-Ott	152986.941

Figura 4: Estratto del dataset con la varianza

Per la soluzione dei job 3 e 4, si è reso necessario andare a fare un calcolo preliminare, ovvero aggregare le informazioni delle vendite in base al mese e all'anno. Successivamente, si è estratto il valore maggiore (per il job 3) e il valore minore (per il job 4) e visto che si è persa l'informazione specifica del mese, è stato necessario andare a recuperarla facendo un merge con la tabella aggregata precedente e le due tabelle risultanti dall'estrazione dei valori di minimo e massimo.

```
1     mesi_max = merge(mesi_max_solo_anno, somma_mesi, by="
    vendite_totali")
```

2.4 MapReduce usando Javascript

Una volta affrontato il problema con lo scritto in R, andiamo ad analizzare il problema sotto il paradigma di MapReduce. Per una prima analisi, utilizziamo

lo strumento del professor Pirrone che permette un rapido approccio a questo modo di scrivere codice. I primi step intrapresi sono andare a dividere il file di input per riga e darlo in pasto allo script di mapping. La fase di mapping prevede la divisione della linea letta in input e sui tre valori che porta (tipologia ordine, data e costo). Successivamente, andiamo (come fatto in precedenza) a filtrare per la tipologia di ordine fattura e ricevuta: nel caso in cui la tipologia d'ordine appartenga a queste due categorie, viene creata una coppia che ha come chiave la data e come valore l'importo, o costo, del documento.

```
    return V_In_Map.map(function(item){
        //scrivo il codice di map
        var tipoOrdine = item.split(",")[0]
        var data = item.split(",")[1]
        var costo = item.split(",")[2]

        var chiave;
        var valore;
        if(tipoOrdine === "FATTURA" ||
        tipoOrdine === "RICEVUTA"){
            chiave = data.slice(0,6);
            valore = costo;
        }
        else
        {
            chiave = "NULL";
            valore = 0;
        }
        return keyVal(chiave, valore);
    });
```

Listing 1: Script di mapping

Serve fare una precisazione: in un paradigma di MapReduce normale, è possibile scartare dei dati in input. In questo tool, è sempre necessario fornire un output di mapping. Di conseguenza, invece di scartare i documenti che non sono utili al fine del programma, vengono mappati con una chiave "null", che nelle elaborazioni successive non verrà considerata.

Per il primo job (Allegato job1.js) , lo script di reduce andrà a recuperare tutti i valori relativi ad una determinata chiave, ovvero ad una data, e a calcolare la media (andandoli a sommare e successivamente a dividere per il numero di valori associati a quella data).

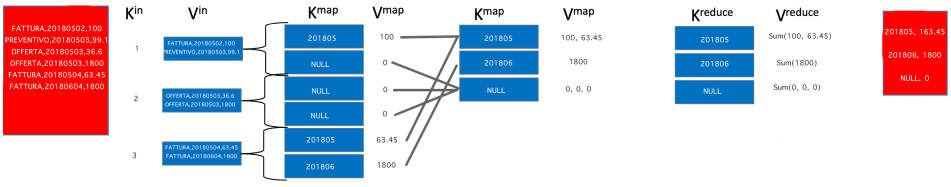

Figura 5: Schema di MapReduce per il primo job

Nel secondo job (Allegato job2.js), andiamo a riciclare lo stesso script di mapping. Per la fase di reduce, andiamo prima a calcolarci la media di valori associati a quella data (esattamente come il job precedente). Il secondo passaggio è quello di andare a calcolarci per ogni valore lo scarto quadratico rispetto alla media, utilizzando la formula (Eq.1, Eq.2). Come ultimo step, ogni valore di scarto quadratico viene sommato alla media.

Per il job 3 e il job 4, risultano necessari due passaggi del processo di MapReduce:

- Il primo passaggio (Allegato prejob34.js) prevede di raccogliere la somma degli importi di ogni mese dell'anno (viene utilizzato parte del codice del primo job senza andare a calcolare la media, ma solo facendo la somma)
- Il secondo passaggio prevede un mapping leggermente diverso, dove come chiave viene inserito l'anno e come valore viene inserito sia il mese che la media degli importi. Lo script di reduce andrà a cercare per ogni anno il valore massimo (per il job 3, Allegato job3.js) e il valore minimo (per il job 4, Allegato job4.js) dell'importo in maniera iterativa. Di conseguenza, questo ha permesso di salvare da parte le informazioni relative al mese e di riportarla a fine elaborazione.

2.5 MapReduce usando R e Hadoop

2.5.1 Scripting

Si è quindi passati ad implementare il MapReduce utilizzando script in R sulla piattaforma Hadoop. Per iniziare ad utilizzare la piattaforma, è fondamentale inizializzare le variabili d'ambiente e far partire i servizi per la gestione delle risorse (Yarn) e il file system condiviso (DFS).

Per l'esecuzione del singolo script, vengono fatte altre operazioni preliminari: per ogni script viene prima eliminata e poi ricreata sul file system condiviso la cartella di lavoro (andando così a pulire le tracce delle precedenti elaborazioni). In seguito, viene caricato all'interno del DFS il file che andremo ad utilizzare come input. Finalmente, possiamo andare a lanciare il nostro script utilizzando lo

streaming di Hadoop. Come ultimo passaggio, andiamo a visualizzare a schermo l'output prodotto, lo copiamo nell'hard disk locale e lo rinominiamo.

Queste operazioni sono state automatizzate da uno script in Bash (Allegato `all_jobs.sh`) e sono state inserite delle chiamate per monitorare le tempistiche e le performance (per salvarle poi in un file).

2.5.2 Implementazione in R

Gli script in R prendono spunto da quelli utilizzati prima nel linguaggio Javascript. Lo script di mapping (Allegato mapper.r), come prima, prende il file in input e per ogni riga, estrae il tipo di ordine, la data e il costo, mandando in output solo le tipologie di ordine presenti nel filtro. Come detto già in precedenza, lo script di mapping è univoco per tutti e quattro i job.

Nel primo script di reduce (Allegato `reducer_job1.r`), andiamo a sfruttare quelle che sono le peculiarità del paradigma di mappare Douce, andando a creare un nuovo Environment. Successivamente, andiamo a leggere l'output dello script precedente, estrarre per ogni riga la data e l'importo, e controllare se all'interno del nuovo ambiente quella data è già stata inserita:

- nel caso il valore della data (mese anno) non fosse presente, viene inserito come importo complessivo l'importo attuale e il conteggio degli elementi viene messo a uno. Il tutto poi viene inserito all'interno dell'ambiente.
- Nel caso il valore se è già presente, viene recuperato l'importo complessivo, al quale si somma l'importo attuale, aumentato il conteggio di uno e inserito il tutto nuovamente all'interno dell'ambiente.

Una volta ultimato questo processo, si passa ad estrarre dall'Environment tutti i valori inseriti e si procede con un processo di "abbellimento" prima di mandare i dati in output.

Il job 2 (Allegato `reducer_job2.r`), nel processo di reduce, ha una peculiarità: oltre ad implementare l'algoritmo per il calcolo della varianza (rispetto a quello di calcolo della media), per recuperare le informazioni relative alle medie, va a leggere all'interno del file system di Hadoop il file di output precedente.

I job 3 e 4 (Allegato `reducer_job3.r` e `reducer_job4.r`), invece, nella fase di reduce, salvano all'interno dell'ambiente la somma degli importi raggruppati per data (mese dell'anno). In un secondo momento, queste informazioni vengono recuperate, aggregate per anno ed estratti i valori di massimo e minimo.

In questa fase, ho preferito non dilungarmi nell'analisi dell'algoritmo (visto che gran parte della complessità si è affrontata con i linguaggi precedenti), ma ho preferito dare risalto a quegli aspetti tipici di questo paradigma di programmazione che Hadoop offre.

2.5.3 MapReduce con Python e Hadoop

Visto che negli ultimi anni Python si sta affermando come linguaggio di programmazione per la data Science, molti programmi e piattaforme si sono aperti alla compatibilità con esso. Su Hadoop, è infatti possibile sfruttare il paradigma MapReduce facendo scripting, oltre che con R, anche con Java e Python. Ho voluto così tradurre il lavoro fatto fino adesso in R anche in Python (Allegato mapper.py, `reducer_job1.py`, `reducer_job2.py`, `reducer_job3.py`, `reducer_job4.py`) e fare una piccola comparazione delle performance di questi due linguaggi. Anche qui, non entro nel merito dell'implementazione, in quanto l'algoritmo è esattamente lo stesso. Cambia solo il linguaggio in cui è scritto. È stato scritto uno script Bash che, a fini statistici, esegue i job di MapReduce circa cento volte e li salva su file.

```
#!/bin/bash
for ((i=1; i<=100; i++))
do
    ./all_jobs.sh
done
```

Listing 2: Script per il salvataggio dei tempi

Successivamente, lo script è stato eseguito sia con uno script R che con uno script Python, salvando i tempi su due file distinti. Successivamente, è stata calcolata la media di esecuzione utilizzando lo strumento JavaScript di MapReduce, come segue:

Tabella 3: Confronto delle Tempistiche tra R e Python per Job

Job	Tempistica R (sec)	Tempistica Python (sec)
1	6.45	5.36
2	8.32	7.21
3	7.54	6.39
4	6.12	5.42
Totale	28.43	24.38

2.6 Rapporto finale

L'analisi delle vendite fornisce una panoramica dettagliata dell'andamento del fatturato dell'azienda con una granularità mensile. Tale analisi può aiutare l'azienda

a comprendere i modelli di vendita, prendere decisioni informate e pianificare per il futuro. [3]

2.6.1 Job 1 - Media di vendita per ogni mese

Dalla Figura 6 si può avere una visione globale delle vendite medie mensili. Tale dato consente di identificare i periodi in cui le vendite sono state particolarmente robuste o al di sotto delle aspettative.

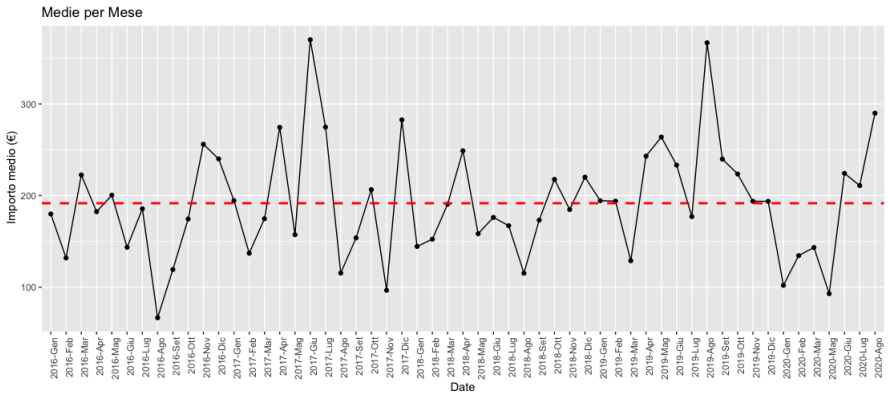

Figura 6: Grafico di elaborazione delle medie mensili

L'andamento di crescita o di stagnazione delle vendite può confermare e influenzare le strategie aziendali e gli sforzi di marketing in determinati mesi e su determinate campagne.

2.6.2 Job 2 - Varianza di vendita per ogni mese

La Figura 7 evidenzia l'ampiezza delle fluttuazioni mensili rispetto alla media. Un valore più alto indica una maggiore volatilità nelle vendite, mentre un valore più basso suggerisce una maggiore stabilità.

[3] È stato notato come in alcuni mesi non sia stato prodotto del fatturato (ad esempio, aprile 2020). Si sarebbe potuto decidere di andare a forzare il valore a 0, ma è stata scelta l'opzione di escludere tale dato poiché è possibile che coincida con un periodo di chiusura aziendale e un valore a 0 non porterebbe informazioni aggiuntive al fine di fare una analisi strategica con le strategie implementate.

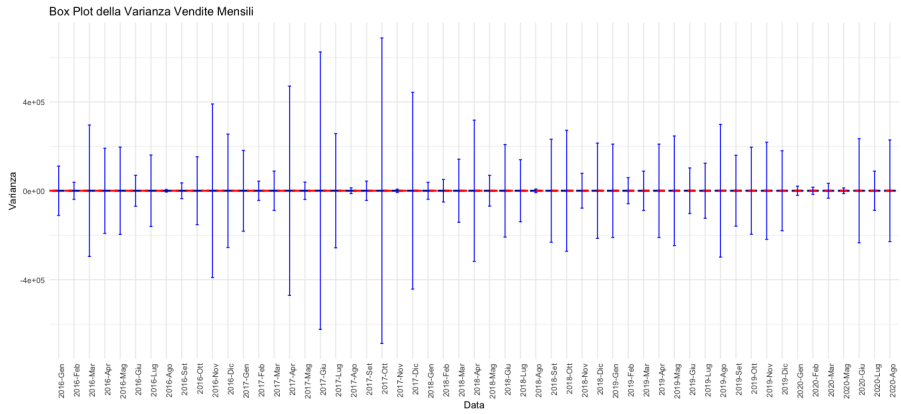

Figura 7: Grafico di elaborazione delle varianze mensili

Queste informazioni sono preziose per comprendere l'incertezza e l'irregolarità delle vendite durante alcuni periodi dell'anno.

2.6.3 Job 3 e 4 - Mesi migliori e peggiori di ogni anno

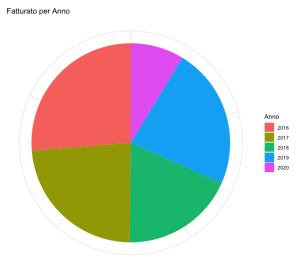

Figura 8: Suddivisione fatturato per anno

La Figura 8 fornisce una visione globale del fatturato dell'azienda in questi anni, contestualizzando i mesi migliori e peggiori di ogni anno (sempre in riferimento alla media globale delle vendite).

L'individuazione del mese con le maggiori vendite (Figura 9) è fondamentale per focalizzare l'attenzione sui fattori che hanno contribuito a questo successo, mentre l'analisi dei mesi con vendite più basse (Figura 10) porta alla luce fattori di debolezza.

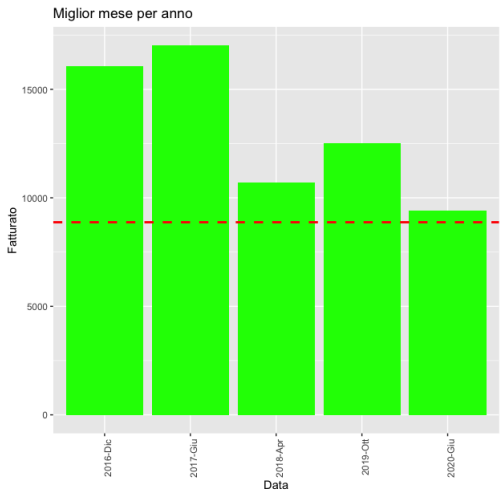

Figura 9: Migliori mesi

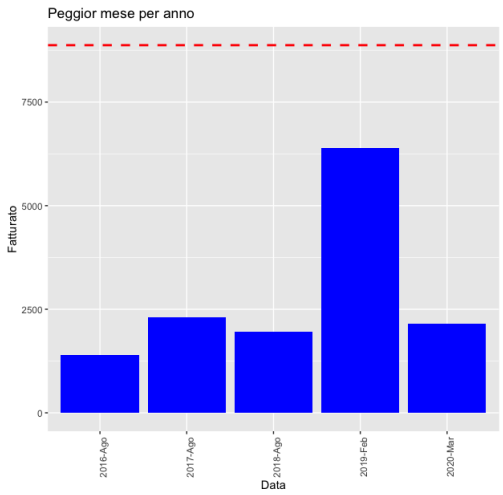

Figura 10: Peggiori mesi

Fattori come la stagionalità, le tendenze di mercato o le concomitanze avverse possono aver influenzato le vendite, così come potrebbero essere state implementate strategie specifiche o eventi particolari che hanno spinto le vendite al di sopra della media.

2.6.4 Analisi

Analizzare le condizioni e le tattiche associate ai mesi in questione può fornire lezioni preziose per guidare le future iniziative aziendali e aiutare a mitigare gli effetti negativi in situazioni simili in futuro.

3 Conclusioni

In questo paper abbiamo visto come l'analisi dei dati sia un processo fondamentale per l'ottimizzazione di qualsiasi attività. Attraverso l'analisi di un dataset relativo al fatturato di un'azienda abbiamo mostrato come sia possibile utilizzare gli strumenti più efficaci per la comprensione dei dati.

In particolare, abbiamo utilizzato R come strumento di analisi dei dati e JavaScript, Hadoop e R per l'implementazione di algoritmi di mapreduce. Grazie all'utilizzo di R, abbiamo creato grafici chiari ed efficaci per la comprensione dei dati, come grafici lineari, box plot e grafici a barre con grafico a torta.

Abbiamo anche illustrato il processo di data science, partendo dall'analisi del problema e del dataset, passando per l'elaborazione di algoritmi di mapreduce fino alla produzione di grafici per rispondere alle necessità del problema iniziale.

Inoltre, abbiamo visto come lo strumento di JavaScript per l'implementazione di algoritmi di mapreduce sia un ottimo strumento didattico per approcciarsi al paradigma di programmazione di mapreduce, soprattutto per chi è alle prime armi e non conosce R o Python. Tuttavia, l'utilizzo di Hadoop come software open source è risultato alquanto complicato da configurare correttamente, soprattutto se si vuole andare ad aggiungere nodi al cluster.

In conclusione, abbiamo apprezzato la possibilità di scalare la propria potenza di calcolo e parallelizzare le informazioni, abbiamo visto come la scelta del linguaggio e delle strategie di implementazione di un algoritmo possano incidere sulle performance e di conseguenza sui costi computazionali di un processo.

La gestione di grandi quantità di dati è una disciplina cruciale in un mondo in cui i dati sono sempre più presenti e utilizzati. Gli strumenti e le metodologie utilizzati in questo documento rappresentano i principi base per l'approccio a questa materia, evidenziando come sia necessario adattarsi alle diverse problematiche proposte.

Allegato

Risoluzione problema tramite R

contabilita.r

```r
path_attuale <- dirname(rstudioapi::getActiveDocumentContext()$path)
setwd(path_attuale)

library(lubridate)
library(dplyr)

#impostazione dati
nomeFile="Ordini.csv"
df=read.csv(nomeFile, sep= ",", header= FALSE)
colnames(df) <- c("tipo","data","costo")
summary(df)
table(df$tipo)
documenti_vendita <- subset(df, tipo=="FATTURA" | tipo=="RICEVUTA")
mesi <- substring(documenti_vendita$data, 1, 6)

#JOB 1
media_mesi <- aggregate(documenti_vendita$costo, by = list(mesi), FUN = mean)
colnames(media_mesi) <- c("data","vendite_medie")
nomi_mesi <- c("Gen", "Feb", "Mar", "Apr", "Mag", "Giu",
               "Lug", "Ago", "Set", "Ott", "Nov", "Dic")
media_mesi <- media_mesi %>%
  mutate(mese = substring(media_mesi$data, 5, 6))
media_mesi$anno <- substr(media_mesi$data, 1, 4)

media_mesi <- media_mesi %>%
  mutate(mese_testuale = nomi_mesi[as.numeric(mese)])

media_mesi <- media_mesi %>%
  mutate(data_formattata = paste(anno, mese_testuale, sep = "-"))

media_mesi <- media_mesi %>%
  select(data ,vendite_medie, data_formattata)

#JOB 2
varianza_mesi = aggregate(documenti_vendita$costo, by = list(mesi), FUN = var)
colnames(varianza_mesi) <- c("data","varianza")
varianza_mesi$anno <- substr(varianza_mesi$data, 1, 4)
```

```
38 varianza_mesi$mese <- substr(varianza_mesi$data, 5, 6)
39 varianza_mesi <- varianza_mesi %>%
40   mutate(mese_testuale = nomi_mesi[as.numeric(mese)])
41 varianza_mesi <- varianza_mesi %>%
42   mutate(data_formattata = paste(anno, mese_testuale, sep = "
     -"))
43 varianza_mesi <- varianza_mesi %>%
44   select(data ,varianza, data_formattata)
45
46 #JOB3
47 somma_mesi = aggregate(documenti_vendita$costo, by = list(
     mesi), FUN = sum)
48 colnames(somma_mesi) <- c("data","vendite_totali")
49 somma_mesi$anno <- substr(somma_mesi$data, 1, 4)
50 anni <- somma_mesi$anno
51 mesi_max_solo_anno = aggregate(somma_mesi$vendite_totali, by
     =list(anni), FUN=max)
52 colnames(mesi_max_solo_anno) <- c("anno_di_riferimento","
     vendite_totali")
53 mesi_max = merge(mesi_max_solo_anno, somma_mesi, by="vendite_
     totali")
54 mesi_max$anno <- substr(mesi_max$data, 1, 4)
55 mesi_max$mese <- substr(mesi_max$data, 5, 6)
56 mesi_max <- mesi_max %>%
57   mutate(mese_testuale = nomi_mesi[as.numeric(mese)])
58 mesi_max <- mesi_max %>%
59   mutate(data_formattata = paste(anno, mese_testuale, sep = "
     -"))
60
61 #JOB 4
62 mesi_min_solo_anno = aggregate(somma_mesi$vendite_totali, by
     =list(anni), FUN=min)
63 colnames(mesi_min_solo_anno) <- c("anno_di_riferimento","
     vendite_totali")
64 mesi_min = merge(mesi_min_solo_anno, somma_mesi, by="vendite_
     totali")
65 mesi_min$anno <- substr(mesi_min$data, 1, 4)
66 mesi_min$mese <- substr(mesi_min$data, 5, 6)
67 mesi_min <- mesi_min %>%
68   mutate(mese_testuale = nomi_mesi[as.numeric(mese)])
69 mesi_min <- mesi_min %>%
70   mutate(data_formattata = paste(anno, mese_testuale, sep = "
     -"))
71
72 #creazione grafici
73 library(ggplot2)
74
75 media_generale <- mean(media_mesi$vendite_medie)
76 media_mesi <- media_mesi[order(media_mesi$data),]
```

```r
media_mesi$data_formattata <- factor(media_mesi$data_formattata, levels = unique(media_mesi$data_formattata))

#medie per mese
ggplot(media_mesi, aes(x = data_formattata, y = vendite_medie, group = 1,)) +
 geom_line() + geom_point() + ggtitle("Medie per Mese") +
  geom_hline(yintercept = media_generale, color = "red", linetype = "dashed", size = 1) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(x = "Date", y = " Importo medio")

#varianza per mese
dati_vendite <- merge(media_mesi, varianza_mesi, by = "data", all = TRUE)
dati_vendite <- dati_vendite[order(dati_vendite$data),]
dati_vendite$data_formattata.x <- factor(dati_vendite$data_formattata.x, levels = unique(dati_vendite$data_formattata.x))
#boxplot varianza
ggplot(dati_vendite, aes(x = data_formattata.x, y = vendite_medie)) +
  geom_boxplot(fill = "lightblue", color = "darkblue") +
  geom_hline(aes(yintercept = mean(vendite_medie)), color = "red", linetype = "dashed", linewidth = 1) +
  geom_errorbar(aes(ymin = vendite_medie - varianza, ymax = vendite_medie + varianza), width = 0.2, color = "blue") +
  labs(title = "Box Plot della Varianza Vendite Mensili",
       x = "Data",
       y = "Varianza") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))

fatturato_per_anno <- aggregate(vendite_totali ~ anno, data = somma_mesi, sum)
# Creazione del grafico a torta
ggplot(fatturato_per_anno, aes(x = "", y = vendite_totali, fill = as.factor(anno))) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar(theta = "y") +
  labs(title = "Fatturato per Anno",
       fill = "Anno") +
  theme_minimal() +
  theme(axis.title = element_blank(),
        axis.text = element_blank(),
        axis.ticks = element_blank(),
        legend.position = "right")
media_generale <- mean(somma_mesi$vendite_totali)
#migliori mesi
```

```
115 ggplot(mesi_max, aes(x = data_formattata, y = vendite_totali)
    ) +
116 geom_bar(stat = "identity", fill = "green", position =
    position_dodge(width = 0.8)) +
117 labs(title = "Miglior mese per anno",
118        x = "Data",
119        y = "Fatturato") + theme(axis.text.x = element_text(
    angle = 90, hjust = 1)) +
120        geom_hline(yintercept = media_generale, color = "red",
     linetype = "dashed", size = 1)
121 #peggiori mesi
122 ggplot(mesi_min, aes(x = data_formattata, y = vendite_totali)
    ) +
123 geom_bar(stat = "identity", fill = "blue", position =
    position_dodge(width = 0.8)) +
124 labs(title = "Peggior mese per anno",
125        x = "Data",
126        y = "Fatturato")  + theme(axis.text.x = element_text(
    angle = 90, hjust = 1)) +
127        geom_hline(yintercept = media_generale, color = "red",
     linetype = "dashed", size = 1)
```

Codice MapReduce

Javascript per Mapreduce Tool

Script inseriti nel Tool Js+html

job1.js

```
1
2  function jobInputSplit ( input_str ) {
3      return input_str.split('\n');
4  }
5  function jobMap ( V_In_Map ) {
6
7      return V_In_Map.map(function(item){
8          //scrivo il codice di map
9          var tipoOrdine = item.split(",")[0]
10         var data = item.split(",")[1]
11         var costo = item.split(",")[2]
12
13         var chiave;
14         var valore;
15         if(tipoOrdine === "FATTURA" || tipoOrdine === "
    RICEVUTA"){
16             chiave = data.slice(0,6);
17             valore = costo;
```

```
        }
        else
        {
            chiave = "NULL";
            valore = 0;
        }
        return keyVal(chiave, valore);
    });
}

function jobReduce ( K_In_Reduce_V_In_Reduce ) {

    return K_In_Reduce_V_In_Reduce.map(function (items){
        var K_In_Reduce = items.split(S[0])[0];
        var V_In_Reduce = items.split(S[0])[1].split(S[1]);
        //scrivo il codice di split
        var Reduce = V_In_Reduce.reduce(function (accumulator
    , item,) {
            if(K_In_Reduce !== "NULL"){
                return parseInt(accumulator) + parseInt(item)
    ;
            }
            return 0;
        },0);
        var k_out = K_In_Reduce;
        var v_out = (Reduce/V_In_Reduce.length).toFixed(2)  ;
        return keyVal (k_out,v_out);
    });
}
```

job2.js[4]

```
function jobReduce ( K_In_Reduce_V_In_Reduce ) {

    return K_In_Reduce_V_In_Reduce.map(function (items){
        var K_In_Reduce = items.split(S[0])[0];
    var V_In_Reduce = items.split(S[0])[1].split(S[1]);
    //scrivo il codice di reduce
    var totale = V_In_Reduce.reduce(function (accumulator,
   item,) {
             return parseInt(accumulator) + parseInt(item);
    },0);
    var media = totale/V_In_Reduce.length;
    console.log(media);
    var scarti_quadratici = V_In_Reduce.map( item =>
          Math.pow(parseInt(media) - parseInt(item),2)
```

[4]NB: per il job2 e il prejob34.js non è riportato lo script di mapping in quanto è uguale a quello del job1

```
    );
    var scarto_quadratico_medio = scarti_quadratici.reduce(
   function (accumulator, item,) {
                return parseInt(accumulator) + parseInt(item)
   ;
    },0)/scarti_quadratici.length;
    console.log(scarto_quadratico_medio);
    var k_out = K_In_Reduce;
        var v_out = scarto_quadratico_medio.toFixed(2) ;
        return keyVal (k_out,v_out);
    });
}
```

prejob34.js[5]

```
function jobReduce ( K_In_Reduce_V_In_Reduce ) {

    return K_In_Reduce_V_In_Reduce.map(function (items){
        var K_In_Reduce = items.split(S[0])[0];
    var V_In_Reduce = items.split(S[0])[1].split(S[1]);
    //scrivo il codice di reduce
    var Reduce = V_In_Reduce.reduce(function (accumulator,
   item,) {
            return parseInt(accumulator) + parseInt(item);
    },0);
    var k_out = K_In_Reduce;
        var v_out = Reduce.toFixed(2);
        return keyVal (k_out,v_out);
    });
}
```

job3.js

```
function jobMap ( V_In_Map ) {

    return V_In_Map.map(function(item){
        //scrivo il codice di map
        var data = item.split("|")[0]
        var anno = data.slice(0,4);
        var mese = data.slice(4,6);
        var media = item.split("|")[1]
        var chiave= anno;
        var valore = mese+"-"+media;
        return keyVal(chiave, valore);
    });
}

```

[5]Questo script è stato utilizzato come pre processazione per i job3 e job4

```
function jobReduce ( K_In_Reduce_V_In_Reduce ) {

    return K_In_Reduce_V_In_Reduce.map(function (items){
        var K_In_Reduce = items.split(S[0])[0];
    var V_In_Reduce = items.split(S[0])[1].split(S[1]);
    //scrivo il codice di reduce
    var mese_massimo;
    var Reduce = V_In_Reduce.reduce(function (massimo, item,)
    {
        var mese_item = item.split("-")[0];
        var media_item = item.split("-")[1];
        if((parseFloat(media_item)>parseFloat(massimo))||
   massimo==0){
            mese_massimo = mese_item;
            return media_item;
        } else {
            return massimo;
        }
    },0);
    var k_out = K_In_Reduce;
        var v_out = mese_massimo+"-"+ Reduce;
        return keyVal (k_out,v_out);
    });
}
```

job4.js

```
function jobMap ( V_In_Map ) {

    return V_In_Map.map(function(item){
        //scrivo il codice di map
        var data = item.split("|")[0]
        var anno = data.slice(0,4);
        var mese = data.slice(4,6);
        var media = item.split("|")[1]
        var chiave= anno;
        var valore = mese+"-"+media;
        return keyVal(chiave, valore);
    });
}

function jobReduce ( K_In_Reduce_V_In_Reduce ) {

    return K_In_Reduce_V_In_Reduce.map(function (items){
        var K_In_Reduce = items.split(S[0])[0];
    var V_In_Reduce = items.split(S[0])[1].split(S[1]);
    //scrivo il codice di reduce
    var mese_minimo;
    var Reduce = V_In_Reduce.reduce(function (minimo, item,)
```

```
    {
23          var mese_item = item.split("-")[0];
24          var media_item = item.split("-")[1];
25          if((parseFloat(media_item)<parseFloat(minimo))||
    minimo==0){
26              mese_minimo = mese_item;
27              return media_item;
28          } else {
29              return minimo;
30          }
31      },0);
32      var k_out = K_In_Reduce;
33          var v_out = mese_minimo+"-"+ Reduce;
34          return keyVal (k_out,v_out);
35      });
36  }
```

Script Bash

all_jobs.sh[6]

```
1   #!/bin/bash
2   export HADOOP_OPTS=-Djava.library.path=/usr/local/hadoop/lib/
        native
3   export HADOOP_HOME=/usr/local/hadoop
4   export HADOOP_CMD=/usr/local/hadoop/bin/hadoop
5   export HADOOP_STREAMING=/usr/local/hadoop/share/hadoop/tools/
        lib/hadoop-streaming-2.6.5.jar
6   export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
7   work_folder=$PWD
8   cd $HADOOP_HOME/sbin
9   start-dfs.sh
10  start-yarn.sh
11
12  cd work_folder
13  rm -f performance.txt
14  hdfs dfs -rm -r /homework
15  hdfs dfs -mkdir /homework
16
17  # Job1
18  hdfs dfs -rm -r /homework/job1
19  hdfs dfs -mkdir /homework/job1
20  hdfs dfs -copyFromLocal $work_folder/Ordini.csv /homework/
        job1
21  rm -f medie_calcolate
```

[6] il codice per eseguire i job in python è stato omesso in quanto è molto simile a quello in R, cambiano solo i nomi degli script in input

```
22 echo "Job1 - Somme iniziato"
23 start_time=$(date +%s)
24 hadoop jar $HADOOP_STREAMING -files $work_folder/mapper_job1.
       r,$work_folder/reducer_job1.r -mapper 'Rscript mapper_job1
       .r' -reducer 'Rscript reducer_job1.r' -input /homework/
       job1/Ordini.csv -output /homework/job1/output
25 end_time=$(date +%s)
26 hdfs dfs -cat /homework/job1/output/part-00000
27 hdfs dfs -copyToLocal /homework/job1/output/part-00000 ./
28 elapsed_time=$((end_time - start_time))
29 echo "Job1 : ${elapsed_time} seconds" >> performance.txt
30 mv ./part-00000 ./medie_calcolate
31 hdfs dfs -mv /homework/job1/output/part-00000 /homework/job1/
       output/medie_calcolate
32
33 # Job2
34 hdfs dfs -rm -r /homework/job2
35 hdfs dfs -mkdir /homework/job2
36 hdfs dfs -copyFromLocal $work_folder/Ordini.csv /homework/
       job2
37 rm -f varianze
38 echo "Job2 - Varianze iniziato"
39 start_time=$(date +%s)
40 hadoop jar $HADOOP_STREAMING -files $work_folder/mapper_job1.
       r,$work_folder/reducer_job2.r -mapper 'Rscript mapper_job1
       .r' -reducer 'Rscript reducer_job2.r' -input /homework/
       job2/Ordini.csv -output /homework/job2/output
41 end_time=$(date +%s)
42 hdfs dfs -cat /homework/job2/output/part-00000
43 hdfs dfs -copyToLocal /homework/job2/output/part-00000 ./
44 elapsed_time=$((end_time - start_time))
45 echo "Job2 : ${elapsed_time} seconds" >> performance.txt
46 mv ./part-00000 ./varianze
47 hdfs dfs -mv /homework/job2/output/part-00000 /homework/job2/
       output/varianze
48
49 # Job3
50 hdfs dfs -rm -r /homework/job3
51 hdfs dfs -mkdir /homework/job3
52 hdfs dfs -copyFromLocal $work_folder/Ordini.csv /homework/
       job3
53 rm -f max.txt
54 echo "Job 3 - Miglior mese per anno"
55 start_time=$(date +%s)
56 hadoop jar $HADOOP_STREAMING -files $work_folder/mapper_job1.
       r,$work_folder/reducer_job3.r -mapper 'Rscript mapper_job1
       .r' -reducer 'Rscript reducer_job3.r' -input /homework/
       job3/Ordini.csv -output /homework/job3/output
57 end_time=$(date +%s)
```

```
hdfs dfs -cat /homework/job3/output/part-00000
hdfs dfs -copyToLocal /homework/job3/output/part-00000 ./
elapsed_time=$((end_time - start_time))
echo "Job3: ${elapsed_time} seconds" >> performance.txt
mv ./part-00000 ./max.txt
hdfs dfs -mv /homework/job3/output/part-00000 /homework/job3/output/max

# Job4
hdfs dfs -rm -r /homework/job4
hdfs dfs -mkdir /homework/job4
hdfs dfs -copyFromLocal $work_folder/Ordini.csv /homework/job4
rm -f min.txt
echo "Job 4 - Peggior mese per anno"
start_time=$(date +%s)
hadoop jar $HADOOP_STREAMING -files $work_folder/mapper_job1.r,$work_folder/reducer_job4.r -mapper 'Rscript mapper_job1.r' -reducer 'Rscript reducer_job4.r' -input /homework/job4/Ordini.csv -output /homework/job4/output
end_time=$(date +%s)
hdfs dfs -cat /homework/job4/output/part-00000
hdfs dfs -copyToLocal /homework/job4/output/part-00000 ./
elapsed_time=$((end_time - start_time))
echo "Job4 : ${elapsed_time} seconds" >> performance.txt
mv ./part-00000 ./min.txt
hdfs dfs -mv /homework/job4/output/part-00000 /homework/job4/output/min

cd $HADOOP_HOME/sbin
stop-all.sh
```

Script R

mapper.r

```
filtro <- c("FATTURA", "RICEVUTA")

connessione <- file("stdin", open = "r")

while (length(line <- readLines(connessione, n = 1, warn = FALSE)) > 0) {
  fields <- unlist(strsplit(line, ","))

  tipoOrdine <- fields[1]
  data <- substr(fields[2], 1, 6)
  costo <- fields[3]

```

```
  if (tipoOrdine %in% filtro){
    cat(data, "\t", costo, "\n", sep = "")
  }
}
close(connessione)
```

reducer_job1.r

```
nomi_mesi <- c("Gen", "Feb", "Mar", "Apr", "Mag", "Giu",
               "Lug", "Ago", "Set", "Ott", "Nov", "Dic")

env <- new.env(hash = TRUE)

connessione <- file("stdin", open = "r")

while (length(line <- readLines(connessione, n = 1, warn =
   FALSE)) > 0) {
  valore_linea <- unlist(strsplit(line, "\t"))
  vendita <- list(mese = valore_linea[1], importo = as.
   numeric(valore_linea[2]))
  risultato <- list(importo  = 0, count = 0)

  if (exists(vendita$mese, envir = env, inherits = FALSE)) {
      valore_corrente <- get(vendita$mese, envir = env)
      risultato$count <- valore_corrente$count + 1
      risultato$importo <- valore_corrente$importo + vendita$
   importo
  }
  else {
    risultato$count <- 1
    risultato$importo <- vendita$importo
  }

  assign(vendita$mese, risultato, envir = env)
}
close(connessione)

for (data in ls(env, all = TRUE)) {
  valore <- get(data, envir = env)
  mese = substring(data, 5, 6)
  anno = substring(data, 1, 4)
  mese_testuale = nomi_mesi[as.numeric(mese)]
  data_formattata = paste(anno, mese_testuale, sep = "-")
    importo = round(valore$importo / valore$count, digits =
   2)
  cat(data, "\t", importo ,"\t",data_formattata, "\n", sep =
   "")
}
```

reducer_job2.r

```r
library(data.table)
nomi_mesi <- c("Gen", "Feb", "Mar", "Apr", "Mag", "Giu",
               "Lug", "Ago", "Set", "Ott", "Nov", "Dic")

nome_file_precedente <- "hadoop fs -text /homework/job1/output/medie_calcolate"
dati_precedenti <- fread(nome_file_precedente)

env <- new.env(hash = TRUE)
connessione <- file("stdin", open = "r")
while (length(line <- readLines(connessione, n = 1, warn = FALSE)) > 0) {

  valore_linea <- unlist(strsplit(line, "\t"))
  vendita <- list(data = valore_linea[1], importo = as.numeric(valore_linea[2]))
  risultato <- list(totale  = 0, count = 0)

  if (exists(vendita$data, envir = env, inherits = FALSE)) {
      valore_corrente <- get(vendita$data, envir = env)

      risultato$count <- valore_corrente$count + 1
      media_del_mese <- dati_precedenti[dati_precedenti$V1 == vendita$data]$V2
      risultato$totale <- (valore_corrente$totale + (vendita$importo - media_del_mese)^2)
  } else {
    risultato$count <- 1
    media_del_mese <- dati_precedenti[dati_precedenti$V1 == vendita$data]$V2
    risultato$totale <- ((vendita$importo - media_del_mese)^2)
  }

  assign(vendita$data, risultato, envir = env)
}
close(connessione)

for (data in ls(env, all = TRUE)) {
  valore <- get(data, envir = env)
  mese = substring(data, 5, 6)
  anno = substring(data, 1, 4)
  mese_testuale = nomi_mesi[as.numeric(mese)]
  data_formattata = paste(anno, mese_testuale, sep = "-")
  varianza = round(valore$totale / valore$count, digits = 2)
  cat(data, "\t", varianza ,"\t",data_formattata, "\n", sep = "")
```

```
}
```

reducer_job3.r

```
library(stringr)
nomi_mesi <- c("Gen", "Feb", "Mar", "Apr", "Mag", "Giu",
               "Lug", "Ago", "Set", "Ott", "Nov", "Dic")

env <- new.env(hash = TRUE)

connessione <- file("stdin", open = "r")

while (length(line <- readLines(connessione, n = 1, warn =
   FALSE)) > 0) {
  valore_linea <- unlist(strsplit(line, "\t"))
  vendita <- list(mese = valore_linea[1], importo = as.
   numeric(valore_linea[2]))
  risultato = 0

  if (exists(vendita$mese, envir = env, inherits = FALSE)) {
      valore_salvato <- get(vendita$mese, envir = env)
      risultato <- valore_salvato + vendita$importo
  }
  else {
    risultato = vendita$importo
  }

  assign(vendita$mese, risultato, envir = env)
}

close(connessione)

chiavi <- ls(env, all = TRUE)
valori <- sapply(chiavi, function(chiave) get(chiave, envir =
    env))
df <- data.frame(Chiave = chiavi, Valore = valori)
df$anno <- substr(df$Chiave, 1, 4)
df$mese <- substr(df$Chiave, 5, 6)
df$mese_testuale <- nomi_mesi[as.numeric(df$mese)]
df$data_formattata <- paste(df$anno, df$mese_testuale, sep =
   "-")

lista_anni = unique(df$anno)
for (year in lista_anni) {
  values_in_year <- df[which(df$anno == year), ]
  max <- values_in_year[which.max(values_in_year$Valore), ]
  cat(max$anno, "\t", max$Valore,"\t", max$data_formattata, "
   \n", sep = "")
}
```

reducer_job4.r

```r
library(stringr)
nomi_mesi <- c("Gen", "Feb", "Mar", "Apr", "Mag", "Giu",
               "Lug", "Ago", "Set", "Ott", "Nov", "Dic")

env <- new.env(hash = TRUE)

connessione <- file("stdin", open = "r")

while (length(line <- readLines(connessione, n = 1, warn =
   FALSE)) > 0) {
  valore_linea <- unlist(strsplit(line, "\t"))
  vendita <- list(mese = valore_linea[1], importo = as.
   numeric(valore_linea[2]))
  risultato = 0

  if (exists(vendita$mese, envir = env, inherits = FALSE)) {
      valore_salvato <- get(vendita$mese, envir = env)
      risultato <- valore_salvato + vendita$importo
  }
  else {
    risultato = vendita$importo
  }

  assign(vendita$mese, risultato, envir = env)
}

close(connessione)

chiavi <- ls(env, all = TRUE)
valori <- sapply(chiavi, function(chiave) get(chiave, envir =
    env))
df <- data.frame(Chiave = chiavi, Valore = valori)
df$anno <- substr(df$Chiave, 1, 4)
df$mese <- substr(df$Chiave, 5, 6)
df$mese_testuale <- nomi_mesi[as.numeric(df$mese)]
df$data_formattata <- paste(df$anno, df$mese_testuale, sep =
   "-")


lista_anni = unique(df$anno)
for (year in lista_anni) {
  values_in_year <- df[which(df$anno == year), ]
  min <- values_in_year[which.min(values_in_year$Valore), ]
  cat(min$anno, "\t", min$Valore,"\t", min$data_formattata, "
   \n", sep = "")
}
```

Script Python

mapper.py

```
filtro = ["FATTURA", "RICEVUTA"]
import sys

for line in sys.stdin:
    fields = line.strip().split(",")

    tipoOrdine = fields[0]
    data = fields[1][:6]
    costo = fields[2]

    if tipoOrdine in filtro:
        print(data + "\t" + costo)
```

reducer_job1.py

```
nomi_mesi = ["Gen", "Feb", "Mar", "Apr", "Mag", "Giu",
             "Lug", "Ago", "Set", "Ott", "Nov", "Dic"]
env = {}
import sys

for line in sys.stdin:
    valore_linea = line.strip().split("\t")
    vendita = {"mese": valore_linea[0], "importo": float(
   valore_linea[1])}
    risultato = {"importo": 0, "count": 0}

    if vendita["mese"] in env:
        valore_corrente = env[vendita["mese"]]
        risultato["count"] = valore_corrente["count"] + 1
        risultato["importo"] = valore_corrente["importo"] +
   vendita["importo"]
    else:
        risultato["count"] = 1
        risultato["importo"] = vendita["importo"]

    env[vendita["mese"]] = risultato

for data, valore in env.items():
    mese = data[4:6]
    anno = data[0:4]
    mese_testuale = nomi_mesi[int(mese) - 1]
    data_formattata = f"{anno}-{mese_testuale}"
    importo = round(valore["importo"] / valore["count"], 2)
    print(f"{data}\t{importo}\t{data_formattata}")
```

reducer_job2.py

```
import subprocess
import pandas as pd

nomi_mesi = ["Gen", "Feb", "Mar", "Apr", "Mag", "Giu",
             "Lug", "Ago", "Set", "Ott", "Nov", "Dic"]

nome_file_precedente = subprocess.check_output(['hadoop', 'fs', '-text', '/homework/job1/output/medie_calcolate']).decode('utf-8')
dati_precedenti = pd.read_csv(nome_file_precedente, sep='\t')

env = {}

while True:
    line = input()
    if not line:
        break

    valore_linea = line.strip().split("\t")
    vendita = {'data': valore_linea[0], 'importo': float(valore_linea[1])}
    risultato = {'totale': 0, 'count': 0}

    if vendita['data'] in env:
        valore_corrente = env[vendita['data']]
        media_del_mese = dati_precedenti[dati_precedenti['data'] == vendita['data']]['media'].values[0]
        risultato['count'] = valore_corrente['count'] + 1
        risultato['totale'] = valore_corrente['totale'] + (vendita['importo'] - dati_precedenti[dati_precedenti['data'] == vendita['data']]['media'].values[0]) ** 2
    else:
        media_del_mese = dati_precedenti[dati_precedenti['data'] == vendita['data']]['media'].values[0]
        risultato['count'] = 1
        risultato['totale'] = (vendita['importo'] - dati_precedenti[dati_precedenti['data'] == vendita['data']]['media'].values[0]) ** 2

    env[vendita['data']] = risultato

for data, valore in env.items():
    mese = data[5:7]
    anno = data[0:4]
    mese_testuale = nomi_mesi[int(mese) - 1]
    data_formattata = f"{anno}-{mese_testuale}"
    varianza = round(valore['totale'] / valore['count'], 2)
```

```
39      print(f"{data}\t{varianza}\t{data_formattata}")
```

reducer_job3.py

```
1  import pandas as pd
2  import sys
3
4  def get_month_name(month):
5      nomi_mesi = ["Gen", "Feb", "Mar", "Apr", "Mag", "Giu",
6                   "Lug", "Ago", "Set", "Ott", "Nov", "Dic"]
7      return nomi_mesi[month - 1]
8
9  env = {}
10
11 for line in sys.stdin:
12     valore_linea = line.strip().split("\t")
13     vendita = {'mese': valore_linea[0], 'importo': float(
       valore_linea[1])}
14     risultato = 0
15
16     if vendita['mese'] in env:
17         valore_salvato = env[vendita['mese']]
18         risultato = valore_salvato + vendita['importo']
19     else:
20         risultato = vendita['importo']
21
22     env[vendita['mese']] = risultato
23
24 chiavi = list(env.keys())
25 valori = list(env.values())
26
27 df = pd.DataFrame({'Chiave': chiavi, 'Valore': valori})
28 df['anno'] = df['Chiave'].str[:4]
29 df['mese'] = df['Chiave'].str[4:6]
30 df['mese_testuale'] = df['mese'].astype(int).apply(
       get_month_name)
31 df['data_formattata'] = df['anno'] + '-' + df['mese_testuale'
       ]
32
33 lista_anni = df['anno'].unique()
34
35 for year in lista_anni:
36     values_in_year = df[df['anno'] == year]
37     max_value = values_in_year.loc[values_in_year['Valore'].
       idxmax()]
38     print(f"{max_value['anno']}\t{max_value['Valore']}\t{
       max_value['data_formattata']}")
```

reducer_job4.py

```
import pandas as pd
import sys

nomi_mesi = ["Gen", "Feb", "Mar", "Apr", "Mag", "Giu",
             "Lug", "Ago", "Set", "Ott", "Nov", "Dic"]
env = {}

for line in sys.stdin:
    valore_linea = line.strip().split("\t")
    vendita = {'mese': valore_linea[0], 'importo': float(
   valore_linea[1])}
    risultato = 0

    if vendita['mese'] in env:
        valore_salvato = env[vendita['mese']]
        risultato = valore_salvato + vendita['importo']
    else:
        risultato = vendita['importo']

    env[vendita['mese']] = risultato

chiavi = list(env.keys())
valori = list(env.values())

df = pd.DataFrame({'Chiave': chiavi, 'Valore': valori})
df['anno'] = df['Chiave'][:4]
df['mese'] = df['Chiave'][4:6]
df['mese_testuale'] = df['mese'].apply(lambda mese: nomi_mesi
   [int(mese) - 1])
df['data_formattata'] = df['anno'] + '-' + df['mese_testuale'
   ]

lista_anni = df['anno'].unique()

for year in lista_anni:
    values_in_year = df[df['anno'] == year]
    min_value = values_in_year.loc[values_in_year['Valore'].
   idxmin()]
    print(f"{min_value['anno']}\t{min_value['Valore']}\t{
   min_value['data_formattata']}")
```

Elenco delle figure

Riferimenti bibliografici

[1] Jeffrey Dean and Sanjay Ghemawa. Mapreduce: Simplified data processing on large clusters. url: https://static.googleusercontent.com/media/research.google.com/it//archive/mapreduce-osdi04.pdf.

[2] Python Software Foundation. Python. url: https://www.python.org/.

[3] The Apache Software Foundation. Apache hadoop. url: https://hadoop.apache.org/.

[4] PBC Posit. Rstudio ide. url: https://www.rstudio.com/categories/rstudio-ide/.

[5] The CRAN team. R archive. url: https://cran.r-project.org/. E-mail: CRAN@R-project.org.

[6] Wikipedia. Python. url: https://it.wikipedia.org/wiki/Python.