Progetto Scalable and Cloud Programming
Co-purchase Analysis

1 Descrizione

L'obiettivo del progetto è quello di realizzare una implementazione in Scala + Spark di un'analisi di co-acquisto di prodotti su un dataset di acquisti. L'analisi di co-acquisto consiste nel calcolare il numero di volte in cui due prodotti fanno entrambi parte di un medesimo ordine di acquisto. In questo modo è possibile analizzare le affinità fra prodotti: per affinità fra due prodotti si intende che se un utente acquista uno dei due prodotti, con alta probabilità acquisterà anche l'altro. Questo tipo di informazione viene spesso usata nei cosiddetti sistemi di raccomandazione.

Al presente documento viene allegato un dataset di acquisti sul quale testare la propria implementazione dell'analisi di co-acquisto. Tale dataset è una versione semplificata del "InstaCart Online Grocery Basket Analysis Dataset" presente sul repository Kaggle (`https://www.kaggle.com/`). Tale dataset contiene un insieme di record di ordini fatto in una app di delivery della spesa al supermercato. Quando dall'app un utente fa un ordine, questo viene inviato ad un rider che si occupa di fare la spesa e consegnare i prodotti a casa dell'utente. Tutti i dati degli ordini vengono raccolti così da poter essere analizzati e capire meglio le abitudini dei clienti.

1.1 Formato del dataset

Ogni record è composto da una coppia (o, p) dove o è l'identificativo dell'ordine e p è l'identificativo del prodotto acquistato. Gli identificativi degli ordini e dei prodotti sono numeri interi non negativi. Se con un singolo ordine si acquistano più prodotti diversi, allora ci sarà un record per ogni prodotto, quindi un ordine con n prodotti genererà n record con l'identificativo dell'ordine ripetuto. Si può assumere che non ci siano ordini vuoti, e che la quantità di un singolo prodotto in un ordine sia sempre 1 (ovvero non ci sono ordini con più quantità dello stesso prodotto).

Il dataset di esempio viene fornito in formato CSV, ovvero un file di testo in cui ogni riga contiene due numeri interi (rispettivamente l'identificativo di un ordine e l'identificativo di un prodotto) separati da virgola. A titolo di esempio, qui sotto viene riportato un possibile dataset:

```
1,12
1,14
2,8
2,12
2,14
3,8
3,12
3,14
3,16
```

contenente tre ordini rispettivamente con identificativo `1`, `2` e `3`: il primo ordine contiene i prodotti `12` e `14`, il secondo i prodotti `8`, `12` e `14` e il terzo i prodotti `8`, `12`, `14` e `16`.

1.2 Formato del risultato dell'analisi

L'analisi di co-acquisto deve calcolare, per ogni coppia di articoli che sono apparsi almeno una volta nel medesimo ordine, il numero di ordini in cui questi articoli appaiono entrambi. Ad esempio, facendo riferimento al semplice dataset riportato nella precedente sezione, si ha che gli articoli `12` e `16` appaiono insieme in un solo ordine, mentre gli articoli `12` e `14` appaiono insieme in due ordini. Il vostro programma Scala+Spark deve effettuare l'analisi di co-acquisto e salvare il risultato in un file in formato CSV che contiene righe nel formato x, y, n dove x e y sono identificatori di due distinti articoli ed n è il numero di ordini in cui x e y appaiono insieme. Se il file contiene la riga x, y, n, non deve essere presente la riga y, x, n visto che l'ordine in cui gli articoli appaiono in un ordine non ha rilevanza.

Nel caso del semplice dataset della precedente sezione, il relativo output conterrebbe le seguenti righe:

```
8,12,2
8,14,2
8,16,1
12,14,3
12,16,1
14,16,1
```

L'ordine in cui appaiono tali righe nel file di output non è rilevante.

1.3 Implementazione

L'elaborazione dei dati deve essere fatta in modo distribuito, deve essere programmata usando l'approccio map-reduce, ed eseguita su piattaforma Apache Spark su DataProc. Si consiglia di provare diversi approcci confrontandone le prestazioni. Per valutare prestazioni e scalabilità bisogna fare diverse prove con cluster di dimensioni differenti.

2 Consegna

Il progetto è individuale e deve essere consegnato entro il 1 Ottobre 2025 (data di scadenza degli education credit di Google Cloud Platform).

La consegna si effettua tramite il sito "virtuale" del corso caricando un report in PDF di massimo **5 pagine** in cui si descrive il lavoro svolto, l'approccio utilizzato, ed una analisi di scalabilità e prestazioni. Nel report va inserito un link ad un repository pubblico su GitHub dove deve essere presente il codice sorgente del progetto con un README contenente le istruzioni per eseguire il programma su DataProc.

Una volta fatta la consegna, bisogna darne comunicazione via e-mail a: `giuseppe.depalma2@unibo.it` e `gianluigi.zavattaro@unibo.it`.

3 Valutazione

La valutazione del progetto è in trentesimi e terrà conto di:

- Approccio utilizzato
- Completezza del report
- Scalabilità
- Prestazioni

4 Comandi Utili

Potete interagire con DataProc sia tramite l'interfaccia web di Google Cloud Platform, sia da terminale. Da terminale è necessario installare il tool **gcloud** e autenticarsi con il proprio account Google (con `gcloud auth login`).

Vengono ora elencati alcuni comandi da terminale che possono essere utili per interagire con DataProc.

Per creare un nuovo cluster DataProc:

```
gcloud dataproc clusters create <nome> --region=<regione>
--num-workers <n> --master-boot-disk-size 240
--worker-boot-disk-size 240
```

In generale si consiglia di impostare un disk size (in questo esempio a 240GB) per evitare di superare il limite imposto agli education credit. Per il numero di workers si consiglia di fare diverse prove con cluster di 1, 2, 3 e 4 workers.

Per immettere un job:

```
gcloud dataproc jobs submit spark --cluster=<nome>
--region=<regione> --jar=gs://<bucket>/<nome-jar>.jar
```

Quando si vuole lanciare il proprio programma su DataProc, potete compilare un JAR e caricarlo su un vostro bucket in Google Cloud Storage precedentemente creato. DataProc è in grado di interagire con i bucket di Google Cloud Storage, quindi basta specificare l'URI appropriato. Questo vale anche per la lettura di dataset (come un file csv) caricato su un bucket e letto via codice Spark tramite l'URI del bucket e infine eseguito su DataProc.

Per cancellare un cluster:
`gcloud dataproc clusters delete <nome> --region <regione>`