Twitch Data Analysis
Relazione per il Secondo Progetto di Big Data

Polosa Sebastiano
498626
seb.polosa@stud.uniroma3.it

Ungaro Riccardo
499606
ric.ungaro@stud.uniroma3.it

A.A. 2020-2021

Indice

Introduzione

In questo elaborato sarà trattato lo svolgimento del secondo progetto di Big Data dove la scelta è stata quella di sviluppare il Topic 4, denominato "ARCHITETTURA LAMBDA", il cui scopo è quello di implementare un'architettura[1] in grado di estrapolare enormi quantità di informazioni, sia in real-time che non, tramite due diversi approcci di analisi: streaming e batch.
Dopo aver presentato il dominio a cui fa parte il dataset utilizzato nella realizzazione del progetto e aver esposto gli obiettivi per cui, secondo noi, c'è necessità di utilizzare questa tipologia di architettura, nella sezione 3 avverrà la trattazione di quali strumenti sono stati utilizzati per la realizzata dell'architettura λ con un particolare focus sui singoli job per entrambe le tipologie di analisi.

1 Caso di studio e obiettivi del progetto

Nelle pagine di questa prima sezione sarà presentato il dominio in cui si è deciso di operare; successivamente avverrà un'analisi del dataset utilizzato per verificare che la nostra struttura di analisi porti ai risultati sperati. Infine saranno argomentati gli obiettivi che si è deciso perseguire con la realizzazione di questo progetto.

1.1 Descrizione del dominio

Il progetto che è stato proposto in questo report ha come scopo quello di effettuare un'analisi batch e streaming su dati raccolti mediante un crawler [1] dalla piattaforma di live-streaming Twitch.tv.
Twitch è una piattaforma online in cui qualsiasi creatore di contenuti multimediali può trasmettere in diretta delle riprese riguardanti una specifica categoria, in genere relative al mondo dei videogiochi, con la possibilità di avere un'interazione, diretta o indiretta, con un pubblico di spettatori.
Effettuare analisi sui dati provenienti da una piattaforma come questa possono portare molteplici benefici in differenti scenari: uno sviluppatore di videogiochi può prendere spunto dai contenuti di maggiore successo per la creazione di nuovi giochi, un creatore di contenuti che si cimenta per la prima volta nella pubblicazione di video su questa piattaforma può osservare quali sono le categorie di maggior successo, una società in cerca di collaborazioni pubblicitarie può prendere delle decisioni sulla base dei dati raccolti. Questi elencati sono solo alcuni esempi dell'utilizzo che si potrebbe fare di un'attenta analisi su questi dati.

1.2 Obiettivi del progetto

Possedere una grande quantità di dati può portare dei vantaggi (economici e/o umani) a chi li possiede e per questo motivo è importante saperli analizzare per estrapolarne l'informazione intrinseca. Nel nostro progetto ci siamo chiesti quale valore aggiunto può portare il possedimento di questi dati estratti dalla piattaforma Twitch e conseguentemente ci siamo posti i seguenti obiettivi:

- scoprire quali sono i contenuti e i broadcaster che più attraggono il pubblico di utenti nella piattaforma;
- fornire ai creatori di contenuti sulla piattaforma degli strumenti utili per effettuare un'analisi sui propri iscritti al canale e sulle proprie live;
- classificare quali sono i giochi più amati dai creatori di contenuti e quali sono quelli più seguiti dagli utenti della piattaforma;

[1] Il nome "architettura lambda" proviene dalla particolare forma che questa possiede che ricorda proprio la lettera greca λ.

- proporre agli utenti che navigano il sito dei contenuti che sono in rilievo.[2]

1.3 Descrizione del dataset

Il dataset[3] impiegato per la realizzazione del progetto utilizza dei dati raccolti ad intervalli regolari dalla piattaforma Twitch mediante un sistema di crawling multi-thread; questo dataset è stato utilizzato per simulare un crawling in real-time di dati da analizzare tramite una serie di strumenti appartenenti ad un'architettura λ (che sarà trattata nello specifico nella sottosezione 3.1).
Il dataset utilizzato in questo progetto è formato da due directory principali:

- **Broadcaster list**: si tratta di una lista dei canali di Twitch su cui è stato effettuato il crawling. Al suo interno sono presenti tre file denominati *all_broadcaster_dict*, *ps4_broadcaster_dict* e *xbox_broadcaster_dict* che riportano rispettivamente l'elenco di tutti gli ID degli streamer che generano contenuti indipendentemente dalla piattaforma utilizzata, un elenco di tutti gli ID dei broadcaster che utilizzano la console di gioco PlayStation per la creazione di contenuti e di tutti quelli che lo fanno tramite la Xbox.

- **Twitch dataset**: questo gruppo di file è il vero e proprio risultato del lavoro svolto dal crawler. Ogni documento è un'insieme di informazioni relative ad una serie di live streaming monitorate dal crawler: per ogni live è possibile sapere a quale categoria appartiene, quando è iniziata, quanti spettatori stanno seguendo attualmente la diretta e quale canale la sta trasmettendo, oltre che ad altre informazioni relative al canale stesso; la Tabella 1 elenca tutte le possibili tipologie di informazioni che sono contenute in questa serie di file. Il crawling è avvenuto ad intervalli regolari di 5 minuti a partire dal primo giorno del mese di febbraio 2015 fino ad arrivare al 7 marzo dello stesso anno, fornendoci una collezione di documenti della durata di poco più di un mese.

2 Strumenti utilizzati

Per lo svolgimento del progetto e la corretta analisi dei dati è stato scelto di utilizzare le seguenti tecnologie:

- **Python**[4]: si tratta di un linguaggio di programmazione orientato agli oggetti molto diffuso nel campo dell'analisi dei dati. Python permette l'utilizzo di potenti librerie di manipolazione dei dati e supporta diversi framework di analisi in ambienti distribuiti.

- **Kafka**[5]: è uno strumento di *event streaming* basato su un sistema publish-subscribe, che si contraddistingue per essere veloce, scalabile e duraturo. Nella sottosottosezione 3.1.1 questo framework è stato trattato più nel dettaglio.

- **MongoDB**[6]: questo è un database NoSQL con struttura di tipo *document*; si è deciso di utilizzare questa tipologia di archiviazione in quanto offre la possibilità di recuperare i dati non solo tramite le chiavi principali ma anche attraverso i campi di ogni oggetto memorizzato.

[2] Questo obiettivo può essere considerato come un punto di partenza per lo sviluppo di un sistema di raccomandazione; il dataset che è stato utilizzato in questo progetto non possiede i dati storici dei vari utenti sulla piattaforma e quindi non permette di effettuare questo tipo di analisi.
[3] Il dataset è reperibile presso una repository GitHub a questo indirizzo `https://clivecast.github.io`.
[4] `https://www.python.org`
[5] `https://kafka.apache.org`
[6] `https://www.mongodb.com`

Campi	Descrizione
stream ID	Un intero indicante l'ID dello streaming (unico per ogni streaming)
current views	Un intero indicante il numero corrente di spettatori
stream created time	Il timestamp indicante il momento di creazione dello streaming
game name	Una stringa indicante il nome della categoria
broadcaster ID	Un intero indicante l'ID del canale (unico per ogni canale)
broadcaster name	Una stringa indicante il nome del canale
delay settings	Un intero indicante le impostazioni di delay del canale
follower number	Un intero indicante il numero di follower del canale
partner status	Un intero indicante il numero dei commenti
broadcaster language	Una stringa indicante la lingua parlata dal canale
total views broadcaster	Un intero indicante il numero di spettatori totale del canale
language	Una stringa indicante la lingua del sito internet del canale
bradcaster's created time	Un timestamp della data in cui è stato creato il canale
playback bitrate	Un float indicante il numero di bitrate per il playback
source resolution	Una stringa indicante la risoluzione della sorgente

Tabella 1: Campi della directory *Twitch Dataset* ottenuti tramite l'esecuzione del crawler sulla piattaforma Twitch.

- **Spark**[7]: è uno dei framework più potenti per effettuare delle operazioni di analisi su dati in ambienti distribuiti. Uno dei più grandi vantaggi per cui si è scelto di utilizzare questo strumento è proprio per la sua velocità: secondo delle misurazioni ufficiali, si contraddistingue per prestazioni di gran lunga superiori a quelle di Hadoop in quanto Spark fa uso di cache in memoria differentemente dai primi sistemi di Big Data che prediligevano l'uso del disco. Apache Spark è composto da un vasto ecosistema (Figura 1) di strumenti e librerie che supportano una varietà di attività diverse che spaziano dall'analisi dei grafi all'apprendimento automatico e all'analisi streaming.
  Tra tutti gli strumenti offerti da Spark abbiamo scelto di utilizzare per questo progetto le seguenti librerie:
  - **Spark Streaming** che dalla sua versione 3.0 permette di eseguire l'analisi utilizzando le *strutture* acquisendo il nome di *Structured Streaming*. Questo strumento offerto da Spark permette di processare i dati in near-real-time in maniera efficiente ed efficace.
  - **Spark SQL** permette di eseguire l'analisi sui dati non strutturati utilizzando un linguaggio molto conosciuto (SQL) e sfruttando tutta la potenza offerta da Spark.

3 Descrizione del lavoro e risultati ottenuti

In questo capitolo, dopo aver presentato l'architettura lambda (3.1) utilizzata per la realizzazione del progetto, sono stati trattati i diversi job di analisi batch (3.2) e streaming (3.3) che sono stati sviluppati per estrarre le informazioni desiderate dalla moltitudine di dati provenienti dalla piattaforma Twitch.

[7] https://spark.apache.org

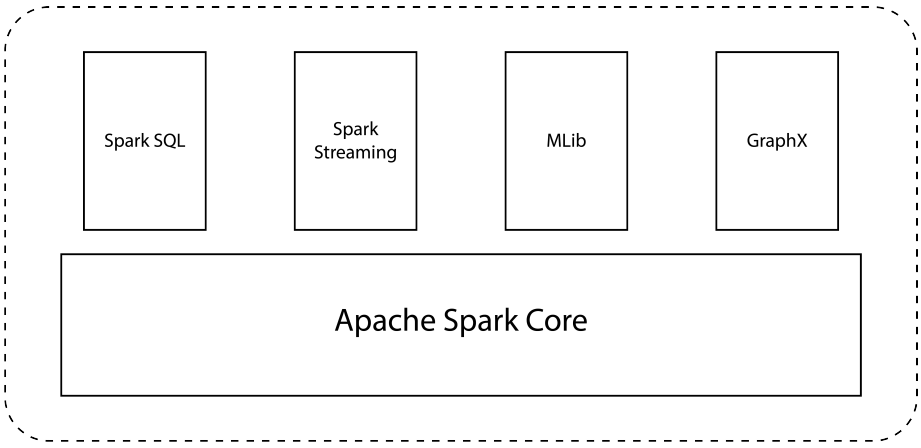


Figura 1: Librerie offerte dal pacchetto di strumenti Spark.

3.1 Descrizione dell'architettura λ

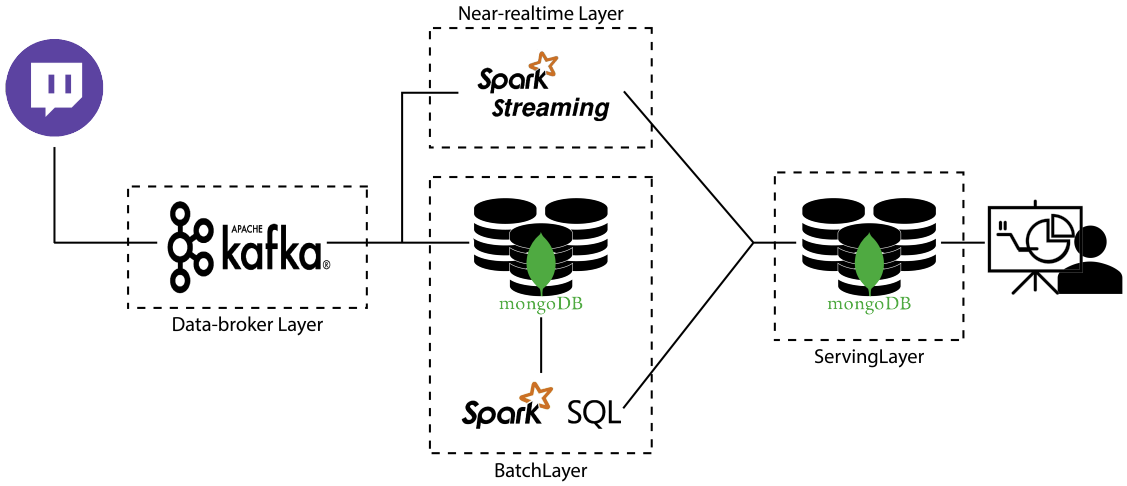
L'architettura lambda è un modello per l'elaborazione dei dati utilizzato per combinare l'analisi batch tradizionale con un'elaborazione di flussi di dati acquisiti in tempo reale per generare un report real time (o near-real-time). Si tratta di un modello di architettura comune per far fronte a enormi volumi di dati.
Nella Figura 2a si può osservare l'adattamento di questa architettura utilizzata nel progetto che è composta da quattro layer principali: *data-broker layer* per il recupero di dati dalle fonti esterne, *batch layer* per l'analisi periodica, *near-real-time layer* per il processamento dei nuovi dati in tempo reale e un *serving layer* per il recupero dei risultati da parte dell'utente .
I dati generati dalla piattaforma sono inviati all'architettura di analisi tramite un sistema di *publish-subscribe* fornito dal framework Kafka; tali dati vengono inviati sia al ramo di analisi streaming, dove sono processati in near-realtime per ottenere informazioni nel più breve tempo possibile, sia al branch di analisi batch. In questo secondo layer, i dati sono dapprima memorizzati all'interno di un DB NoSQL gestito tramite MongoDB e successivamente analizzati ad intervalli periodici tramite uno strumento di analisi batch.
Al termine di ogni analisi, le informazioni estrapolate dai dati sono memorizzate all'interno di un database MongoDB che ha lo scopo di serving layer per le interrogazioni degli utenti che vogliono osservare i risultati ottenuti.

3.1.1 Kafka

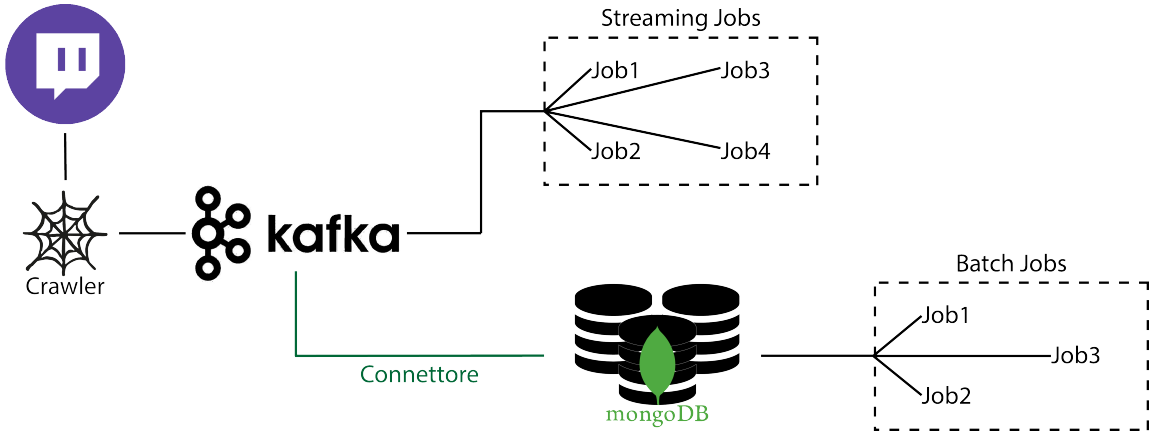
Un ruolo fondamentale all'interno di questa architettura è rivestito dal framework Kafka.

> *Apache Kafka is an open-source distributed event streaming platform used by thousands of companies for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications.*

Per poter comprendere meglio la definizione qui sopra riportata è necessario chiarire cosa si intende per piattaforma di event streaming; effettuare uno *streaming di eventi* significa catturare i dati in real-time dalle varie sorgenti di eventi (database, sensori, dispositivi mobili, applicazioni, ...) per poterli memorizzare, manipolare ed inviare verso tutte le tecnologie che li necessitano. Kafka è uno strumento che permette di effettuare event streaming in ambiente distribuito tramite un sistema di messaggi tra publishers (le sorgenti)



(a) Architettura λ utilizzata per la realizzazione del progetto.



(b) Utilizzo di Kafka nell'architettura del progetto.

Figura 2

e consumers (chi necessita dei dati) e un sistema di topics; inoltre è possibile rendere i dati persistenti per renderli accessibili anche in futuro.
Nella Figura 2b è possibile osservare come Kafka sia stato utilizzato in questo progetto per catturare i dati provenienti dalla piattaforma Twitch ed inviarli ai rispettivi consumer per effettuare le analisi streaming.
Per effettuare l'analisi batch non è necessario processare i dati in tempo reale quindi non occorre effettuare un collegamento diretto tra Kafka e i jobs ma bisogna interporre un database tra i due con il ruolo di data lake. Poiché un database non consuma i dati streaming, il collegamento con Kafka avviene tramite un connettore personalizzato che ne permette il salvataggio persistente così che il db possa essere provvisto di tutte le informazioni quando è necessario che venga interrogato da un job batch in esecuzione.
Sebbene Twitch abbia delle API pubbliche per accedere e scaricare alcune tipologie di informazioni, per semplicità è stato deciso di utilizzare un crawler [1] già implementato e reperibile su internet che, in questo ambiente, svolge il ruolo di publisher.

3.2 Analisi Batch

Il Batch Layer si occupa dell'analisi di set di dati di grandi dimensioni, i quali vengono raccolti per un periodo di tempo (giorni, settimane o mesi) prima di essere processati ed elaborati. In questo metodo di analisi si effettua una suddivisione tra la fase di raccolta dei dati e la fase di processamento.
Le tecnologie proposte per l'elaborazione batch in questo progetto sono MongoDB e Spark: il primo è utilizzato per archiviare nel database *data_lake* i dati inviati da kafka i quali saranno successivamente elaborati dai diversi job proposti, Spark è un'insieme di librerie che è stato utilizzato per l'elaborazione batch.
Tra gli strumenti disponibili con Spark si è deciso di utilizzare *SparkSQL* che è un tool che permette di lavorare con dati strutturati in maniera simile a quello che si fa nell'interazione con un database relazionale. Il core di Spark SQL è la struttura dati chiamata DataFrame: un'importante struttura dati che va oltre gli RDD e rappresenta una collezione immutabile simile ad una tabella di un database relazionale con righe e colonne.

3.2.1 Job 1 - Le piattaforme preferite dai creatori di contenuti

Con il passare degli anni Twitch ha ospitato un sempre più vasto numero di contenuti particolarmente diversi tra loro, ma è bene non dimenticarsi che il focus principale della piattaforma sono stati sin dall'inizio i videogiochi. In questo contesto si è pensato che fosse utile conoscere ed analizzare il numero di utenti che effettua dirette streaming e qual è la piattaforma più apprezzata, infatti è possibile approcciarsi alla piattaforma utilizzando diversi dispositivi come ad esempio un pc o tramite una console (ps4 o xbox).
Il job1 opera a partite da dati contenuti in tre differenti file di testo denominati *all_broadcaster_dict*, *ps4_broadcaster_dict* e *xbox_broadcaster_dict* che contengono ognuno un elenco di ID di broadcaster divisi per piattaforma di utilizzo. Affinché questi dati possano essere caricati sul database utilizzato per il progetto è stato necessario trasformare il formato di questi tre file da *.txt* in *.json* tramite l'utilizzo dello script python *upload_data_to_mongo.py* contenuto nella cartella del job1 stesso. Questo programma legge riga per riga ognuno dei tre file di testo contenenti gli Id dei broadcaster e li archivia in un dizionario Python sotto forma di coppia chiave/valore (esempio: *'broadcaster_id':description*). Questi dizionari sono successivamente salvati su MongoDB nel database *data_lake* in tre collezioni distinte:

- **all**: contenente gli Id di tutti i broadcaster;
- **ps**: contenete gli Id dei broadcaster che usano la piattaforma Playstaion per la creazione dei loro contenuti;
- **xbox**: contenete gli Id dei broadcaster che usano la piattaforma xbox.

all_broadcaster	ps4_broadcaster	xbox_broadcaster	pc_broadcaster
2388705	702705	426483	1259517

Tabella 2: Risultati ottenuti tramite l'esecuzione del job 1 che genera un resoconto di quanti sono gli streamer per ogni piattaforma.

Una volta create le tre collezioni su MongoDB il job1 viene avviato e come prima operazione preleva i dati da queste tramite l'utilizzo di PyMongo, il quale permette di stabilire una connessione con il database utilizzando la classe MongoClient. A partire da questi dati saranno creati tre DataFrame, uno per ogni collezione, sui quali sarà effettuato il calcolo per produrre l'output desiderato. La Tabella 2 mostra i risultati al termine dell'esecuzione del programma.

3.2.2 Job 2 - Il gioco preferito dagli iscritti

L'obiettivo di questo job è quello di produrre un resoconto contenente, per ogni Streamer, il gioco più seguito dai propri iscritti sulla base delle visualizzazioni totali mensili.
Questa analisi ha lo scopo di aiutare i creatori di contenuti della piattaforma a tener traccia di quali giochi preferiscono i propri iscritti mostrando, per ogni streamer, il titolo del gioco che nel mese corrente ha avuto più visualizzazioni totali e consigliando quali siano gli streaming che potrebbero aver maggior successo in futuro.
Per la realizzazione di questo job è stato necessario stabilire una connessione con MongoDB, precisamente con il database *data_lake* e la collezione *twitch* da cui leggere i dati, al fine di recuperare i campi utili per questa analisi: *['current_view','game_name','broadcaster_id','broadcaster_name']*.
Grazie al metodo *spark.createDataFrame()* viene creato un DataFrame Spark sul quale verranno eseguite le operazioni ed interrogazioni per poter produrre la classifica desiderata. La Tabella 3 mostra una parte della classifica generata.

3.2.3 Job 3 - Top 25 Games and Streamer

Si è deciso di implementare questa analisi per produrre tre resoconti a cadenza mensile utilizzabili sia dai creatori di contenuti che dai produttori di videogiochi:

- **"top 25 dei giochi con più contenuti (streaming) sulla piattaforma nel mese d'interesse"**: questa classifica mostra i giochi con più live su Twitch, classificati in base agli streaming totali creati sulla piattaforma negli ultimi 30 giorni;
- **"top 25 giochi più seguiti del mese"**: la top 25 dei giochi più visti su Twitch che presenta i migliori giochi sulla piattaforma classificati in base alle visualizzazioni totali;
- **"top 25 streamer più seguiti del mese"**: quest'ultimo elenco contiene i primi 25 canali con il maggior numero di visualizzazioni sulla piattaforma social.

Per la creazione di queste tre classifiche è stato creato un DataFrame, contenente i dati recuperati dalla collezione *twitch* del database *data_lake* caricato precedentemente su MongoDB, con i seguenti campi: *['current_view','game_name','broadcaster_id','broadcaster_name']*. Successivamente il DataFrame è stato interrogato con tre diverse queries per la produzione delle classifiche desiderate di cui è possibile osservarne degli esempi nelle Tabelle 4a, 4b e 4c a pagina 10.

broadcasterID	broadcasterName	gameName	max(currentViews)
29578325	beyondthesummit	Dota 2	34846
30220059	esltv_sc2	StarCraft II: Hea...	27293
24954143	dotacinema	Dota 2	24142
29795919	nl_kripp	Hearthstone: Hero...	18725
28633266	starladder3	Dota 2	17817
28036688	trick2g	League of Legends	17044
1518077	goldglove	Dying Light	15252
36794584	riotgames2	League of Legends	14314
28633177	starladder1	Dota 2	13070
7951350	cryaotic	Left 4 Dead 2	12037
28633298	starladder4	Dota 2	11039
31478096	mym_alkapone	Gaming Talk Shows	10315
29769280	beyondthesummit2	Dota 2	10148
32803072	bestrivenna	League of Legends	5880
54706574	theoriginalweed	Counter-Strike: G...	5113
38881685	flosd	Dying Light	5080
14293484	voyboy	League of Legends	4832

Tabella 3: Risultati ottenuti tramite l'esecuzione del job 2 che genera un resoconto dei giochi preferite dai propri iscritti sulla base delle visualizzazioni totali mensili

gameName	count
LeagueofLegends	1514
DyingLight	1096
Destiny	519
1	472
CounterStrike:G...	443
Minecraft	418
CallofDuty:Adv...	394
GrandTheftAutoV	369
WorldofWarcraft...	253
H1Z1	235
Dota2	225
CallofDuty:Ad...	225
Battlefield4	177
MaddenNFL15	170
Hearthstone:Hero...	167
NBA2K15	165
FIFA15	164
HeroesoftheStorm	131
Smite	119
DayZ	91

(a) top 25 dei giochi con più streaming sulla piattaforma nel mese d'interesse

gameName	sum(currentViews)
Dota2	118968
LeagueofLegends	68964
StarCraftII:Hea...	32536
DyingLight	28404
Hearthstone:Hero...	28213
CounterStrike:G...	18311
GamingTalkShows	16824
Left4Dead2	12162
WorldofWarcraft...	11699
Minecraft	10004
H1Z1	8027
TheBindingofIs...	5086
Destiny	4753
XCOM:EnemyWithin	4294
CallofDuty:Adv...	4290
RuneScape	3487
GrandTheftAutoV	3472
ResidentEvilArc...	3359
Magic:TheGathering	2952
HeroesoftheStorm	2731

(b) top 25 giochi più seguiti del mese

broadcasterName	sum(currentViews)
beyondthesummit	34846
esltv_sc2	27293
dotacinema	24142
nl_kripp	18725
starladder3	17817
trick2g	17044
goldglove	15252
riotgames2	14314
starladder1	13070
cryaotic	12037
starladder4	11039
mym_alkapone	10315
beyondthesummit2	10148
bestrivenna	5880
theoriginalweed	5113
flosd	5080
voyboy	4832
tsm_theoddone	4282
phantoml0rd	4137
swifty	3984

(c) top 25 streamer più seguiti del mese

Tabella 4: Un esempio delle tre classifiche prodotte per il mese di febbraio.

3.3 Analisi Streaming

Nei paragrafi che seguono verrà trattato come sono stati implementati i diversi jobs per eseguire l'analisi near-realtime sui dati provenienti dalla piattaforma Steam.
Il paradigma che è stato utilizzato in questa fase di analisi prevede l'utilizzo di due tipologie di script per job:

- la prima sfrutta *Spark Streaming* per effettuare una semplificazione dei dati che arrivano dalle fonti con lo scopo di estrapolare le informazioni di interesse;
- la seconda utilizza *Spark SQL* per interrogare gli esiti dello script precedente (salvati in formato .csv) e salvarli su un database del Serving Layer così da renderli accessibili a chiunque.

3.3.1 Job 1 - Ranking by views

La prima tipologia di analisi che sarà esaminata in questo paragrafo prevede di effettuare una classifica ordinata sulla base del numero di visualizzazioni correnti degli streaming attualmente in live.
Per poter adempire a questo compito è stato necessario estrapolare dai dati di input alcune informazioni: l'id univoco per ogni live, il nome del gioco di cui si sta facendo la live, il numero di utenti che stanno visualizzando la live nel momento in cui è stato eseguito il crawl e il timestamp in cui è stato eseguito il crawl; ottenute ciò, queste informazioni vengono memorizzate su un database MongoDB così da renderle persistenti. L'Algoritmo 1 mostra lo pseudocodice di questo primo metodo affinché possa essere replicato.
Un'analisi come questa, eseguita in realtime, potrebbe permettere di generare una classifica da presentare agli utenti indecisi che navigano sul sito per facilitarli nella scelta di quali contenuti potrebbero guardare.
Un esempio di possibili risultati ottenuti eseguendo questo metodo di analisi sul dataset in esame è osservabile nella Tabella 5.

Algoritmo 1: Ranking by views

```
1 while dati in streaming continuano ad arrivare do
2 |   Crea una tabella partendo dai dati appena arrivati con i seguenti campi: stream_id, game_name,
  |     current_view, crawl_time;
3 |   if i record sono gli ultimi ad essere stati recuperati then
4 |   |   Ordina i record in ordine decrescente di current_view;
5 |   |   Salva i dati nel Serving Layer partizionando per orario di calcolo dell'informazione;
6 |   end
7 end
```

3.3.2 Job 2 - Ranking by mean of views

L'idea alla base di questo secondo metodo di analisi nasce durante l'implementazione del Job 1 - Ranking by views con lo scopo di ottenere un altro punto di vista, con una visione d'insieme, delle stesse informazioni. L'obiettivo di questo compito è quello di osservare quali dirette streaming hanno avuto maggior successo durante tutta la loro durata e non solo nell'istante del crawling.
Nell'Algoritmo 2 è possibile leggere lo pseudocodice d'implementazione per questo metodo di analisi. I dati in input trasportati tramite Kafka vengono letti per estrapolarne solo alcuni campi: l'id univoco per ogni live, il nome del gioco di cui si sta facendo la live, il numero di utenti che stanno visualizzando la live nel momento in cui è stato eseguito il crawl, il timestamp della data e ora di creazione dello streaming e il

stream_id	game_name	current_view	crawl_time
12932973168	Dota2	29816	2015-02-0101:15:00
12932549648	StarCraftII:Hea...	27819	2015-02-0101:15:00
12932994272	Dota2	24315	2015-02-0101:15:00
12935159760	Hearthstone:Hero...	19256	2015-02-0101:15:00
12933578608	Dota2	16604	2015-02-0101:15:00
12934530544	LeagueofLegends	16579	2015-02-0101:15:00
12932518304	DyingLighnds	14044	2015-02-0101:15:00
12933966224	DyingLight	11317	2015-02-0101:15:00
12935517168	GamingTalkShows	10138	2015-02-0101:15:00
12935229856	Dota2	8863	2015-02-0101:15:00
12936016272	Dota2	8635	2015-02-0101:15:00
12933757376	Dota2	8447	2015-02-0101:15:00
12932065776	LeagueofLegends	8187	2015-02-0101:15:00
12936030864	LeagueofLegends	6528	2015-02-0101:15:00
12933993312	Counter-Strike:G...	4920	2015-02-0101:15:00
12931778816	DyingLight	4810	2015-02-0101:15:00

Tabella 5: Risultati ottenuti tramite l'analisi eseguita nel Job 1 che prevede di generare una classifica basata sul numero delle visualizzazioni correnti degli streaming attualmente in live.

timestamp in cui è stato eseguito il crawl. Partendo da queste informazioni, il primo step da fare è quello di recuperare l'elenco delle live attive nelle ultime 24 ore[8] dal momento dell'ultimo crawl; poi si procede con il conteggio del numero di occorrenze e la sommatoria di tutte le visualizzazioni per ogni stream_id per poi infine calcolare la media delle views. L'ultimo passo prevede la memorizzazione delle informazioni recuperate su un database del Serving Layer. La Tabella 6 mostra un esempio dei risultati e dei calcoli intermedi eseguiti tramite questo job.
Questo metodo, come il precedente, può essere utilizzato per consigliare agli utenti quale video guardare, con la differenza che può consigliare anche delle live streaming già terminate (avendo una visione temporale più larga) con lo scopo di mostrare ad un utente dei contenuti interessanti di cui potrebbe essersi perso la diretta.

3.3.3 Job 3 - Trend games

Questa tipologia di analisi propone di determinare il numero degli streaming attivi per ogni categoria (identificate tramite il campo *game_name*) con lo scopo di calcolare quali siano quelle maggiormente in tendenza. Come si evince dall'Algoritmo 3, per poter calcolare questo tipo di informazione, per prima cosa è necessario filtrare i dati ottenuti in input prelevando solamente i campi relativi al nome della categoria (*game_name*) e il timestamp in cui è stato eseguito il crawl di tali dati. Successivamente si procede con il calcolo del numero di occorrenze di un determinato nome di una categoria nella tabella dei nuovi record; per poter eseguire questa operazione si può ricorrere ad un *groupBy* basato sul campo *game_name*. La Tabella 7 mostra due output calcolati sui dati del dataset in istanti differenti.

[8] Si è deciso di porre un limite a 24 ore dal recupero delle ultime informazioni per non mostrare informazioni troppo vecchie in quanto lo scopo di questo job è dare informazioni sempre recenti.

Algoritmo 2: Ranking by mean of views

while *dati in streaming continuano ad arrivare* **do**
    Crea una tabella partendo dai dati appena arrivati con i seguenti campi: stream_id, game_name, current_view, stream_created_time, crawl_time;
    Ottieni i record relativi agli streaming delle ultime 24 ore;
    Conta il numero di occorrenze di ogni streaming tramite il campo stream_id;
    Calcola il valore medio delle visualizzazioni per ogni stream_id nelle ultime 24 ore;
    Ordina i risultati ottenuti in ordine decrescente di numero di visualizzazioni medio;
    Salva i dati nel Serving Layer partizionando per orario di calcolo dell'informazione;
end

stream_id	game_name	sum(current_view)	count	average
12933966224	DyingLight	44721	1	44721.0
12932973168	Dota2	159885	5	31977.0
12932549648	StarCraftII:Hea...	140787	5	28157.4
12932994272	Dota2	101718	5	20343.6
12935159760	Hearthstone:Hero...	96551	5	19310.2
12932518304	DyingLight	79886	5	15977.2
12931574736	LeagueofLegends	76945	5	15389.0
12934530544	LeagueofLegends	75330	5	15066.0
12933578608	Dota2	68002	5	13600.4
12935229856	Dota2	61658	5	12331.6
12933966224	Left4Dead2	12037	1	12037.0
12933966224	DyingLight	44721	4	11180.25
12933757376	Dota2	52196	5	10439.2
12936016272	Dota2	52185	5	10437.0
12935517168	GamingTalkShows	51254	5	10250.8
12932065776	LeagueofLegends	37136	5	7427.2
12936030864	LeagueofLegends	30005	5	6001.0

Tabella 6: Risultati ottenuti tramite l'analisi eseguita nel Job 2 che prevede di generare una classifica basata sul numero medio delle visualizzazioni correnti degli streaming che sono stati in live nelle ultime 24 ore.

Come nei casi precedenti, un possibile campo di utilizzo di questo metodo può essere quello di indirizzare un utente alla scelta di quale categoria di video guardare oppure quello di effettuare delle analisi di mercato con lo scopo di investire sulle categorie maggiormente popolari in quel periodo.

Algoritmo 3: Trend games

```
1 while dati in streaming continuano ad arrivare do
2     Crea una tabella partendo dai dati appena arrivati con i seguenti campi: game_name,
        crawl_time;
3     if i record sono gli ultimi ad essere stati recuperati then
4         Conta il numero di occorrenze di ogni game_name;
5         Ordina i risultati ottenuti in ordine decrescente di numero di occorrenze per categoria;
6         Salva i dati nel Serving Layer partizionando per orario di calcolo dell'informazione;
7         Salva i dati nel Serving Layer partizionando per orario di calcolo dell'informazione;
8     end
9 end
```

3.3.4 Job 4 - Views percentage

L'ultimo metodo per l'analisi streaming si pone come obiettivo quello di determinare quale sia la percentuale degli iscritti ad un canale che stanno guardando la live in corso. Per comprendere meglio lo scopo di questo job procediamo con un breve esempio: ipotizziamo che un canale denominato Acme abbia 1526 iscritti e che decida di avviare una diretta streaming. Effettuando un monitoraggio ci rendiamo conto che in media 216 utenti hanno seguito questa live, ovvero solo il 15% degli iscritti. Qualche giorno dopo lo stesso canale decide di effettuare un'altra live dove però partecipano 2435 spettatori con una percentuale spettatori/iscritti = 159%. Tenere sotto controllo questa percentuale potrebbe permettere alla piattaforma di individuare questo tipo di anomalie in modo rapido ed efficace (un altro esempio è visibile nelle prime righe della Tabella 8 dove si può osservare che un canale con soli 2 iscritti ha una percentuale di visualizzazione della live del 4150% e questo può essere uno spunto per effettuare delle indagini più approfondite).
L'implementazione (Algoritmo 4) prevede, per prima cosa, l'estrazione dai dati iniziali dei campi di interesse: un codice univoco della live, il codice identificativo e il nome del canale, il numero dei follower, il numero di spettatori corrente, il nome del gioco e il timestamp in cui è avvenuto il crawl di queste informazioni; successivamente si prosegue con il calcolo della percentuale $view\_percentage = current\_view/follower\_number$. I risultati ottenuti sono poi memorizzati su un database del Serving Layer così da renderli accessibili.
Nella Tabella 8 a pagina 16 è possibile osservare un esempio ordinato dei risultati ottenuti tramite questo metodo di analisi applicato al dataset di riferimento.

game_name	count
League of Legends	1451
Dying Light	991
Destiny	468
Counter-Strike: G...	427
Minecraft	393
Call of Duty: Adv...	345
Grand Theft Auto V	341
World of Warcraft...	239
Call of Duty: Ad...	223
H1Z1	214
Dota 2	211
FIFA 15	187
Battlefield 4	164
Madden NFL 15	159
Hearthstone: Hero...	155
NBA 2K15	150
Heroes of the Storm	113

(a)

game_name	count
League of Legends	1429
Dying Light	952
Destiny	463
Counter-Strike: G...	418
Minecraft	389
Grand Theft AutoV	349
Call of Duty: Adv...	346
World of Warcraft...	234
Call of Duty: Ad...	219
Dota 2	219
H1Z1	209
FIFA 15	190
Battlefield 4	157
Hearthstone: Hero...	157
NBA 2K15	151
Madden NFL 15	150
Heroes of the Storm	122

(b)

Tabella 7: Due esempi di risultati ottenuti in istanti diversi tramite l'analisi eseguita nel Job 3; questa prevede di creare una classifica dei giochi più streammati dai canali basandosi su quelli attualmente in live.

Algoritmo 4: Views percentage

```
1 while dati in streaming continuano ad arrivare do
2     Crea una tabella partendo dai dati appena arrivati con i seguenti campi: stream_id, game_name,
        current_view, broadcaster_id, broadcaster_name, follower_number, crawl_time;
3     Aggiungi una nuova colonna denominata view_percentage;
4     Ordina i risultati ottenuti in ordine decrescente di numero di visualizzazioni percentuali;
5     Assegna col(view_percentage) = col('current_view')/col('follower_number');
6 end
```

stream_id	game_name	current_view	broadcaster_id	broadcaster_name	follower_number	view_percentage
12935516256	Battlefield™Hard...	83	80759669	scooby3751	2	41.5
12931831792	CallofDuty:Adv...	119	37894502	thecodman11	3	39.666668
12935582464	DyingLight	307	55920369	based_gordeez	10	30.7
12936224880	APBReloaded	49	81487127	animebhopper	4	12.25
12935070496	WorldofTanks	21	75422343	xumpbiu_jiuc	2	10.5
12934335120	LeagueofLegends	89	50688254	loyaltylol	9	9.888889
12929952208	Minecraft	397	80702628	thearcalypse	49	8.102041
12935912144	RuneScape	8	81030935	brandirex	1	8.0
12936076544	FF14	8	52916159	lyrisbrue	1	8.0
12936452688	Battlefield™Hard...	15	81497008	youngturk44	2	7.5
12930780752	null	14	79847857	redstone10th	2	7.0
12936372256	LordsoftheFallen	7	51470823	loosdevil	1	7.0
12936173456	ProEvolutionSoc...	6	78260104	davidrebirth77	1	6.0
12935698288	Fibbage:TheHila...	6	45841918	jpeeper	1	6.0
12935775888	FIFA15	6	68116639	the_hawkz	1	6.0
12936128656	LIMBO	6	80823315	thirsty4chicken	1	6.0
12935838880	Raven'scry	18	40520823	frolik12	3	6.0
12936261232	WorldofTanks	5	80633741	dokeid_	1	5.0
12936396352	GrandTheftAuto:...	5	65224056	frank207	1	5.0
12936230960	LifeIsStrange™	5	52468541	vindicatednoxus	1	5.0

Tabella 8: Risultati ottenuti tramite l'analisi eseguita nel Job 4 che prevede di generare una classifica basata sul numero percentuali delle visualizzazioni correnti dello streaming rispetto al numero di iscritti del canale

4 Conclusioni e progetti futuri

Aver scelto di svolgere questa tipologia di progetto ci ha permesso di apprendere il funzionamento di una delle architetture maggiormente utilizzate nel mondo dei Big Data; la decisione di scegliere di affrontare questo Topic è stata guidata dalla curiosità di toccare con mano propria un'architettura di cui abbiamo sentito parlare fin dai primi giorni del corso di Big Data e di Advanced Topics in Computer Science.
Aver avuto la possibilità di approfondire il funzionamento di quelle tecnologie che le aziende utilizzano per effettuare l'analisi di grandi quantità di dati è stato considerato da noi un valore aggiunto non indifferente: abbiamo trovato interessante e divertente capire come questi framework siano in grado di interagire e collaborare tra loro per uno scopo comune.
Tenendo conto del fatto che i risultati ottenuti siano frutto di una nostra esperienza alle prime armi in questo settore, siamo soddisfatti di essere riusciti a implementare un'architettura di analisi dei dati in maniera totalmente automatica e funzionante e vogliamo proporre alcune possibili idee per degli sviluppi futuri:

- si potrebbe considerare l'idea di creare un'interfaccia web per la visualizzazione dei risultati delle analisi;
- per favorire la riproducibilità e la portabilità dell'architettura si potrebbe creare una Docker Image;
- si potrebbe aumentare il numero dei job di analisi (sia streaming che batch) in quanto i dati a disposizione contengono una grande quantità di informazione;
  - un possibile nuovo job streaming potrebbe essere quello di monitorare per ciascun canale se il numero degli iscritti aumenta o diminuisce durante una live così da capire quanto questa influisca sul comportamento degli utenti;
  - un'interessante aggiunta ai job batch potrebbe essere rappresentata dall'analisi del numero di streaming giornalieri e mensili che vengono avviati utilizzando un pc e quanti invece vengono creati utilizzando direttamente una console (ps4 o xbox).
- si consideri l'idea di implementare un secondo crawler per il recupero delle informazioni degli utenti-spettatori delle live così da poter effettuare delle analisi più approfondite;
- raccogliendo una quantità di dati sufficientemente grande è possibile implementare dei metodi di predizione sui trend tramite librerie di machine learning quali per esempio MLib.

Queste appena elencate sono solo alcune possibili idee in un mondo che offre infinite possibilità di analisi (purché si posseggano dati di buona qualità).

Riferimenti bibliografici

[1] Cong Zhang e Jiangchuan Liu. *On Crowdsourced Interactive Live Streaming: A Twitch.TV-Based Measurement Study*. 2015. arXiv: 1502.04666 [cs.MM].