Definizione ed Implementazione di un Sistema di Raccomandazione Distribuito per film e Modellazione di Eventi Complessi

Logica ed Intelligenza Artificiale
Ingegneria del Software Avanzata

Mauro Losciale, Pietro Tedeschi
Prof. Ing. Tommaso Di Noia
Prof.ssa Marina Mongiello
Dott. Francesco Nocera

Politecnico di Bari
Dipartimento di Ingegneria Elettrica e dell'Informazione
SisInf Lab

Indice

Indice

Indice

Indice

Indice

Indice

Indice

Indice

Indice

Indice

Indice

Indice

Indice

Indice

Indice

Introduzione al Progetto

Scopo del Progetto

Costruire un Sistema di Raccomandazione distribuito,utilizzabile con un'interfaccia web, che sia in grado di analizzare lo stream dati in input, gestendo gli eventi complessi mediante query continue sul flusso informativo.

Obiettivo

Il sistema di raccomandazione consentirà di votare un numero di film che sia strettamente maggiore di 3, selezionando i contenuti votati con un rate superiore o uguale a quattro, al fine di raccomandare i film non ancora visti, che si avvicinano di più ai gusti dell'utente.

Tecnologie e Linguaggi

Apache Spark, Apache Kafka, Node.js,Angular.js; Scala, Javascript

Introduzione al Progetto

Scopo del Progetto

Costruire un Sistema di Raccomandazione distribuito,utilizzabile con un'interfaccia web, che sia in grado di analizzare lo stream dati in input, gestendo gli eventi complessi mediante query continue sul flusso informativo.

Obiettivo

Il sistema di raccomandazione consentirà di votare un numero di film che sia strettamente maggiore di 3, selezionando i contenuti votati con un rate superiore o uguale a quattro, al fine di raccomandare i film non ancora visti, che si avvicinano di più ai gusti dell'utente.

Tecnologie e Linguaggi

Apache Spark, Apache Kafka, Node.js,Angular.js; Scala, Javascript

Introduzione al Progetto

Scopo del Progetto

Costruire un Sistema di Raccomandazione distribuito,utilizzabile con un'interfaccia web, che sia in grado di analizzare lo stream dati in input, gestendo gli eventi complessi mediante query continue sul flusso informativo.

Obiettivo

Il sistema di raccomandazione consentirà di votare un numero di film che sia strettamente maggiore di 3, selezionando i contenuti votati con un rate superiore o uguale a quattro, al fine di raccomandare i film non ancora visti, che si avvicinano di più ai gusti dell'utente.

Tecnologie e Linguaggi

Apache Spark, Apache Kafka, Node.js,Angular.js; Scala, Javascript

I Sistemi CEP

Definizione

Un sistema Complex Event Processing (CEP), modella il flusso informativo dei dati, visualizzando gli elementi come notifiche di ciò che sta accadendo nel mondo esterno.

Obiettivo

Identificare EVENTI significativi e rispondere ad essi nel più breve tempo possibile.

Processing Rules

Un pattern o una regola, può essere definita mediante un linguaggio basato su query, il cosiddetto EVENT QUERY LANGUAGE.

I Sistemi CEP

Definizione

Un sistema Complex Event Processing (CEP), modella il flusso informativo dei dati, visualizzando gli elementi come notifiche di ciò che sta accadendo nel mondo esterno.

Obiettivo

Identificare EVENTI significativi e rispondere ad essi nel più breve tempo possibile.

Processing Rules

Un pattern o una regola, può essere definita mediante un linguaggio basato su query, il cosiddetto Event Query Language.

I Sistemi CEP

Definizione

Un sistema Complex Event Processing (CEP), modella il flusso informativo dei dati, visualizzando gli elementi come notifiche di ciò che sta accadendo nel mondo esterno.

Obiettivo

Identificare EVENTI significativi e rispondere ad essi nel più breve tempo possibile.

Processing Rules

Un pattern o una regola, può essere definita mediante un linguaggio basato su query, il cosiddetto Event Query Language.

Architettura di un sistema CEP

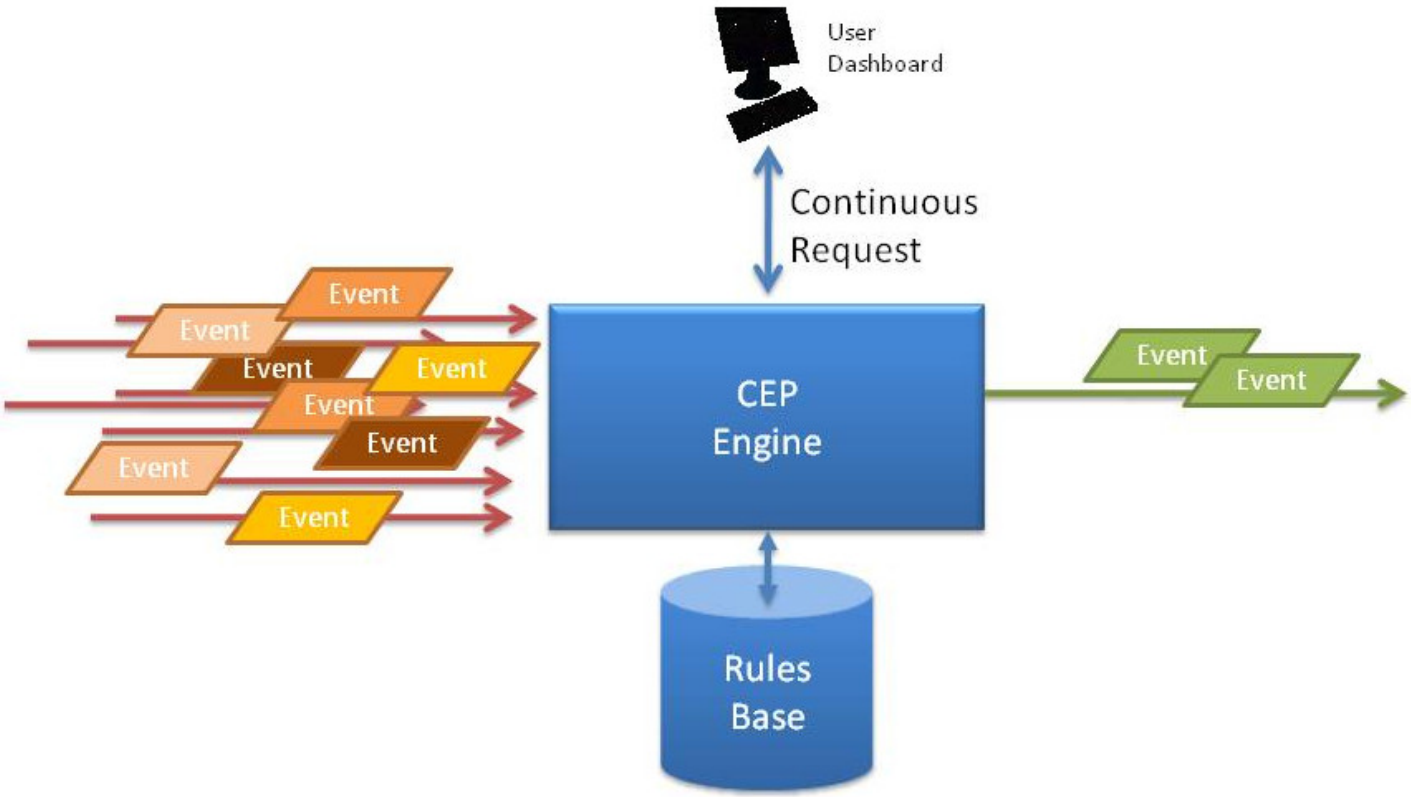


Figura: Architettura di un Sistema CEP

I Sistemi di Raccomandazione

Definizione

I sistemi di raccomandazione, raccolgono informazioni sulle preferenze di utente in corrispondenza di un insieme di elementi. L'informazione può essere:

- **Implicita**
- **Esplicita**

Si usano per fornire una miglior **Quality of Experience** all'utente.

Esempi

Amazon, Netflix, Google, Last.fm, Facebook, Twitter, MovieLens, ecc.

Classificazione algoritmi di Filtraggio

- *Collaborative* Filtering
- *Demographic Filtering*
- *Content-Based Filtering*
- *Hybrid Filtering*

I Sistemi di Raccomandazione

Definizione

I sistemi di raccomandazione, raccolgono informazioni sulle preferenze di utente in corrispondenza di un insieme di elementi. L'informazione può essere:

- **Implicita**
- **Esplicita**

Si usano per fornire una miglior **Quality of Experience** all'utente.

Esempi

Amazon, Netflix, Google, Last.fm, Facebook, Twitter, MovieLens, ecc.

Classificazione algoritmi di Filtraggio

- *Collaborative* Filtering
- *Demographic Filtering*
- *Content-Based Filtering*
- *Hybrid Filtering*

I Sistemi di Raccomandazione

Definizione

I sistemi di raccomandazione, raccolgono informazioni sulle preferenze di utente in corrispondenza di un insieme di elementi. L'informazione può essere:

- **Implicita**
- **Esplicita**

Si usano per fornire una miglior **Quality of Experience** all'utente.

Esempi

Amazon, Netflix, Google, Last.fm, Facebook, Twitter, MovieLens, ecc.

Classificazione algoritmi di Filtraggio

- *Collaborative* Filtering
- *Demographic Filtering*
- *Content-Based Filtering*
- *Hybrid Filtering*

I Sistemi di Raccomandazione

Definizione

I sistemi di raccomandazione, raccolgono informazioni sulle preferenze di utente in corrispondenza di un insieme di elementi. L'informazione può essere:

- **Implicita**
- **Esplicita**

Si usano per fornire una miglior **Quality of Experience** all'utente.

Esempi

Amazon, Netflix, Google, Last.fm, Facebook, Twitter, MovieLens, ecc.

Classificazione algoritmi di Filtraggio

- *Collaborative Filtering*
- *Demographic Filtering*
- *Content-Based Filtering*
- *Hybrid Filtering*

I Sistemi di Raccomandazione

Definizione

I sistemi di raccomandazione, raccolgono informazioni sulle preferenze di utente in corrispondenza di un insieme di elementi. L'informazione può essere:

- **Implicita**
- **Esplicita**

Si usano per fornire una miglior **Quality of Experience** all'utente.

Esempi

Amazon, Netflix, Google, Last.fm, Facebook, Twitter, MovieLens, ecc.

Classificazione algoritmi di Filtraggio

- *Collaborative* Filtering
- *Demographic Filtering*
- *Content-Based Filtering*
- *Hybrid Filtering*

I Sistemi di Raccomandazione

Definizione

I sistemi di raccomandazione, raccolgono informazioni sulle preferenze di utente in corrispondenza di un insieme di elementi. L'informazione può essere:

- **Implicita**
- **Esplicita**

Si usano per fornire una miglior **Quality of Experience** all'utente.

Esempi

Amazon, Netflix, Google, Last.fm, Facebook, Twitter, MovieLens, ecc.

Classificazione algoritmi di Filtraggio

- *Collaborative* Filtering
- *Demographic Filtering*
- *Content-Based Filtering*
- *Hybrid Filtering*

I Sistemi di Raccomandazione

Definizione

I sistemi di raccomandazione, raccolgono informazioni sulle preferenze di utente in corrispondenza di un insieme di elementi. L'informazione può essere:

- **Implicita**
- **Esplicita**

Si usano per fornire una miglior **Quality of Experience** all'utente.

Esempi

Amazon, Netflix, Google, Last.fm, Facebook, Twitter, MovieLens, ecc.

Classificazione algoritmi di Filtraggio

- *Collaborative* Filtering
- *Demographic Filtering*
- *Content-Based Filtering*
- *Hybrid Filtering*

I Sistemi di Raccomandazione

Definizione

I sistemi di raccomandazione, raccolgono informazioni sulle preferenze di utente in corrispondenza di un insieme di elementi. L'informazione può essere:

- **Implicita**
- **Esplicita**

Si usano per fornire una miglior **Quality of Experience** all'utente.

Esempi

Amazon, Netflix, Google, Last.fm, Facebook, Twitter, MovieLens, ecc.

Classificazione algoritmi di Filtraggio

- *Collaborative* Filtering
- *Demographic Filtering*
- *Content-Based Filtering*
- *Hybrid Filtering*

I Sistemi di Raccomandazione

Definizione

I sistemi di raccomandazione, raccolgono informazioni sulle preferenze di utente in corrispondenza di un insieme di elementi. L'informazione può essere:

- **Implicita**
- **Esplicita**

Si usano per fornire una miglior **Quality of Experience** all'utente.

Esempi

Amazon, Netflix, Google, Last.fm, Facebook, Twitter, MovieLens, ecc.

Classificazione algoritmi di Filtraggio

- *Collaborative* Filtering
- *Demographic Filtering*
- *Content-Based Filtering*
- *Hybrid Filtering*

I Sistemi di Raccomandazione

Il processo con cui un sistema di raccomandazione generi una raccomandazione, è basato sulla combinazione delle seguenti considerazioni:

Passi per la Raccomandazione

1. Tipologia di dati disponibili nel database.
2. Algoritmo di filtraggio usato.
3. Modello scelto '**memory-based**' o '**model-based**').
4. Tecniche impiegate: approccio probabilistico, reti Bayesiane, algoritmi genetici,. . . .
5. Livello di dispersione del database e scalabilità desiderata.
6. Capacità di elaborazione del sistema (tempo di elaborazione e consumi di memoria).
7. Obiettivo da raggiungere (predizioni e raccomandazioni)
8. Qualità del risultato desiderata.

I Sistemi di Raccomandazione

Il processo con cui un sistema di raccomandazione generi una raccomandazione, è basato sulla combinazione delle seguenti considerazioni:

Passi per la Raccomandazione

1. Tipologia di dati disponibili nel database.
2. Algoritmo di filtraggio usato.
3. Modello scelto '**memory-based**' o '**model-based**').
4. Tecniche impiegate: approccio probabilistico, reti Bayesiane, algoritmi genetici,. . . .
5. Livello di dispersione del database e scalabilità desiderata.
6. Capacità di elaborazione del sistema (tempo di elaborazione e consumi di memoria).
7. Obiettivo da raggiungere (predizioni e raccomandazioni)
8. Qualità del risultato desiderata.

I Sistemi di Raccomandazione

Il processo con cui un sistema di raccomandazione generi una raccomandazione, è basato sulla combinazione delle seguenti considerazioni:

Passi per la Raccomandazione

1. Tipologia di dati disponibili nel database.
2. Algoritmo di filtraggio usato.
3. Modello scelto '**memory-based**' o '**model-based**').
4. Tecniche impiegate: approccio probabilistico, reti Bayesiane, algoritmi genetici,. . . .
5. Livello di dispersione del database e scalabilità desiderata.
6. Capacità di elaborazione del sistema (tempo di elaborazione e consumi di memoria).
7. Obiettivo da raggiungere (predizioni e raccomandazioni)
8. Qualità del risultato desiderata.

I Sistemi di Raccomandazione

Il processo con cui un sistema di raccomandazione generi una raccomandazione, è basato sulla combinazione delle seguenti considerazioni:

Passi per la Raccomandazione

1. Tipologia di dati disponibili nel database.
2. Algoritmo di filtraggio usato.
3. Modello scelto '**memory-based**' o '**model-based**').
4. Tecniche impiegate: approccio probabilistico, reti Bayesiane, algoritmi genetici,. . . .
5. Livello di dispersione del database e scalabilità desiderata.
6. Capacità di elaborazione del sistema (tempo di elaborazione e consumi di memoria).
7. Obiettivo da raggiungere (predizioni e raccomandazioni)
8. Qualità del risultato desiderata.

I Sistemi di Raccomandazione

Il processo con cui un sistema di raccomandazione generi una raccomandazione, è basato sulla combinazione delle seguenti considerazioni:

Passi per la Raccomandazione

1. Tipologia di dati disponibili nel database.
2. Algoritmo di filtraggio usato.
3. Modello scelto '**memory-based**' o '**model-based**').
4. Tecniche impiegate: approccio probabilistico, reti Bayesiane, algoritmi genetici,....
5. Livello di dispersione del database e scalabilità desiderata.
6. Capacità di elaborazione del sistema (tempo di elaborazione e consumi di memoria).
7. Obiettivo da raggiungere (predizioni e raccomandazioni)
8. Qualità del risultato desiderata.

I Sistemi di Raccomandazione

Il processo con cui un sistema di raccomandazione generi una raccomandazione, è basato sulla combinazione delle seguenti considerazioni:

Passi per la Raccomandazione

1. Tipologia di dati disponibili nel database.
2. Algoritmo di filtraggio usato.
3. Modello scelto '**memory-based**' o '**model-based**').
4. Tecniche impiegate: approccio probabilistico, reti Bayesiane, algoritmi genetici,....
5. Livello di dispersione del database e scalabilità desiderata.
6. Capacità di elaborazione del sistema (tempo di elaborazione e consumi di memoria).
7. Obiettivo da raggiungere (predizioni e raccomandazioni)
8. Qualità del risultato desiderata.

I Sistemi di Raccomandazione

Il processo con cui un sistema di raccomandazione generi una raccomandazione, è basato sulla combinazione delle seguenti considerazioni:

Passi per la Raccomandazione

1. Tipologia di dati disponibili nel database.
2. Algoritmo di filtraggio usato.
3. Modello scelto '**memory-based**' o '**model-based**').
4. Tecniche impiegate: approccio probabilistico, reti Bayesiane, algoritmi genetici,....
5. Livello di dispersione del database e scalabilità desiderata.
6. Capacità di elaborazione del sistema (tempo di elaborazione e consumi di memoria).
7. Obiettivo da raggiungere (predizioni e raccomandazioni)
8. Qualità del risultato desiderata.

I Sistemi di Raccomandazione

Il processo con cui un sistema di raccomandazione generi una raccomandazione, è basato sulla combinazione delle seguenti considerazioni:

Passi per la Raccomandazione

1. Tipologia di dati disponibili nel database.
2. Algoritmo di filtraggio usato.
3. Modello scelto '**memory-based**' o '**model-based**').
4. Tecniche impiegate: approccio probabilistico, reti Bayesiane, algoritmi genetici,....
5. Livello di dispersione del database e scalabilità desiderata.
6. Capacità di elaborazione del sistema (tempo di elaborazione e consumi di memoria).
7. Obiettivo da raggiungere (predizioni e raccomandazioni)
8. Qualità del risultato desiderata.

I Sistemi di Raccomandazione

Il processo con cui un sistema di raccomandazione generi una raccomandazione, è basato sulla combinazione delle seguenti considerazioni:

Passi per la Raccomandazione

1. Tipologia di dati disponibili nel database.
2. Algoritmo di filtraggio usato.
3. Modello scelto '**memory-based**' o '**model-based**').
4. Tecniche impiegate: approccio probabilistico, reti Bayesiane, algoritmi genetici,....
5. Livello di dispersione del database e scalabilità desiderata.
6. Capacità di elaborazione del sistema (tempo di elaborazione e consumi di memoria).
7. Obiettivo da raggiungere (predizioni e raccomandazioni)
8. Qualità del risultato desiderata.

Collaborative Filtering

Il **Collaborative Filtering** consente agli utenti di attribuire un voto ad un insieme di elementi salvando le proprie preferenze all'interno di un database, e consentendo di creare una raccomandazione specifica per ogni utente.

Similarità

- **item-item**: due elementi sono simili se tendono ad ottenere lo stesso rate dagli utenti.
- **user-user**: due utenti sono simili se tendono a dare lo stesso rate agli elementi.

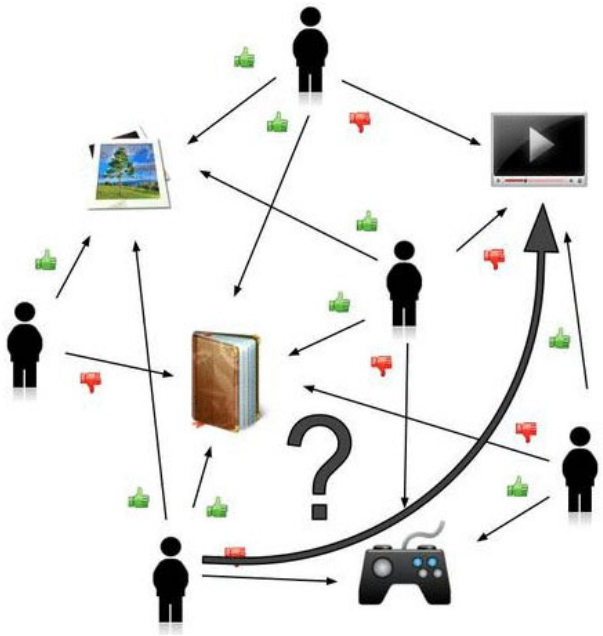


Figura: Collaborative Filtering

Root Mean Squared Error

La qualità di un sistema di raccomandazione può essere valutata dai risultati forniti in output. Le tipologie di metriche usate, dipendono dal tipo di CF utilizzato.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i,j}(p_{i,j} - r_{i,j})^2}$$

Definizione

Nella seguente formula, n è il numero totale dei voti assegnati dagli utenti, $p_{i,j}$ è il voto predetto per l'utente i sull'elemento j, ed $r_{i,j}$ è il voto attuale. L'**RMSE** consente di valutare in maniera precisa i contributi degli errori assoluti tra i valori predetti ed i valori reali.

Matrix Factorization

Definizione

I modelli di **Matrix Factorization** possono essere utilizzati per ricercare quali sono i fattori latenti che caratterizzano le interazioni tra due tipologie di entità.

- U:insieme degli utenti.
- D:insieme degli elementi.
- R:matrice dei voti di dimensioni $|U| \times |D|$.

Passi

Supponiamo di voler cercare i K fattori latenti.

- Cercare P, una matrice $|U| \times K$.
- Cercare Q, una matrice $|D| \times K$.
- Il prodotto di tali matrici sia una stima di $R \approx P \times Q^T = \hat{R}$.

$$\hat{r_{i,j}} = {p_i}^T q_j = \sum_{1}^{k} p_{ik} q_{kj}$$

Algoritmi di minimizzazione dell'errore

SGD: Stochastic Gradient Descent - **ALS**: Alternating Least Squares

Definizione

Data Stream, una sequenza di elementi codificati in digitale rappresentanti un'informazione. Al fine di gestire un flusso dati continuo, sono nati i **Data Stream Management System** (DSMS).

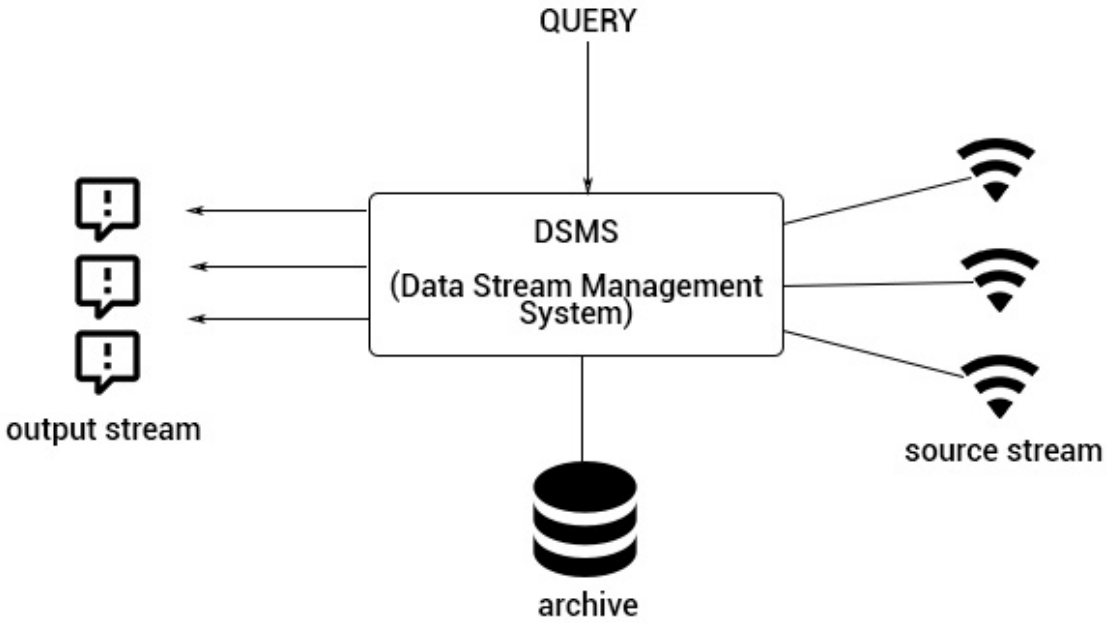


Figura: Architettura DSMS

Data Stream Processing II

Caratteristiche

1. Data-Driven
2. Elevata flessibilità nell'eseguire Query continue sullo Stream dati
3. I dati arrivano online
4. L'ordine di arrivo degli elementi non è importante
5. Nessuna dimensione prefissata

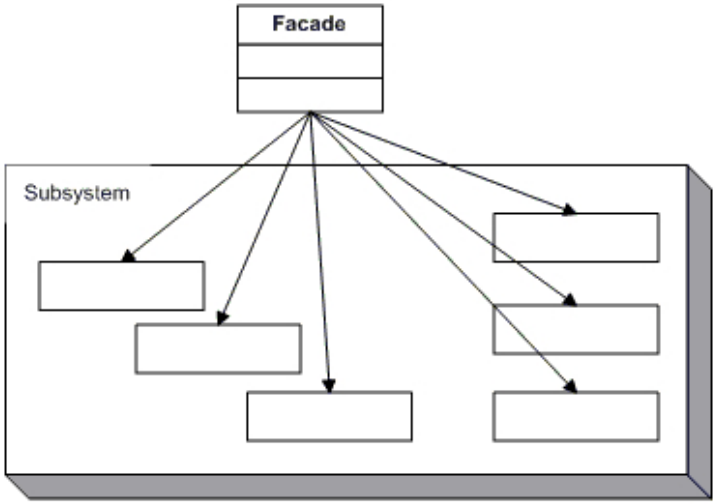


Figura: Pattern Facade

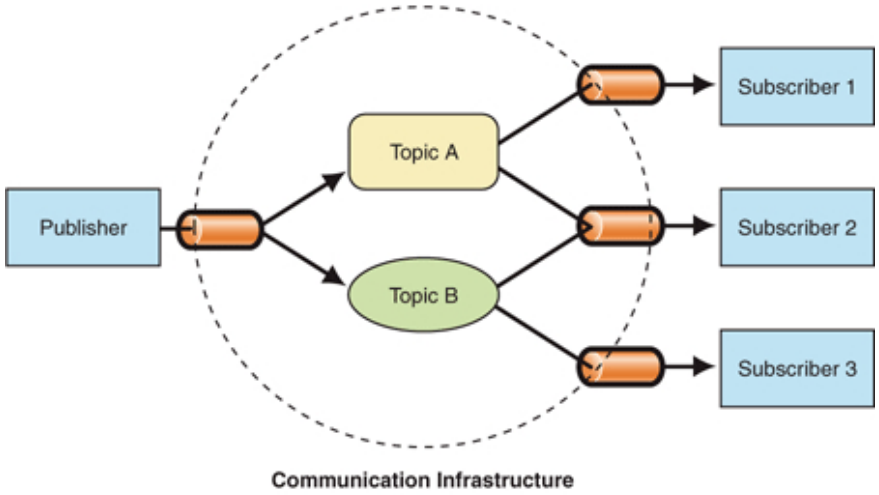


Figura: Publish/Subscribe

Figura: Pattern Singleton

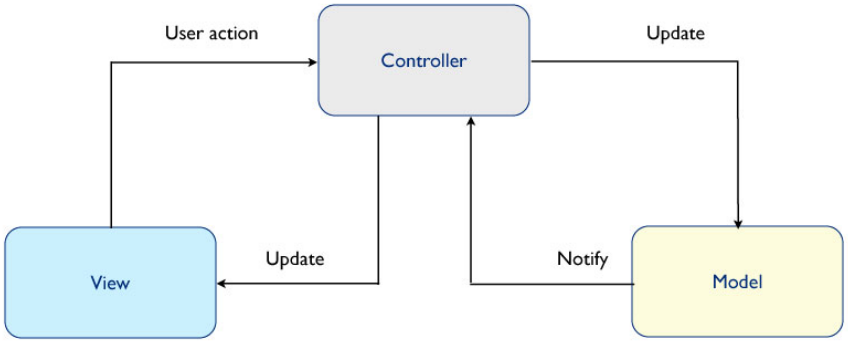


Figura: Pattern Model View Controller

Definizione e Caratteristiche

Il protocollo **WebSocket** consente la comunicazione bidirezionale e full duplex mediante una singola connessione TCP.

- Connessione persistente instaurata mediante un sistema di handshaking client-key.
- Sicurezza origin-based.
- Porte di comunicazione, 80 e 443.
- Schema URI utilizzato è `ws://` per connessioni in chiaro, `wss://` per connessioni cifrate con TLS.
- Comunicazione a bassa latenza, efficiente e quasi realtime tra client e server.

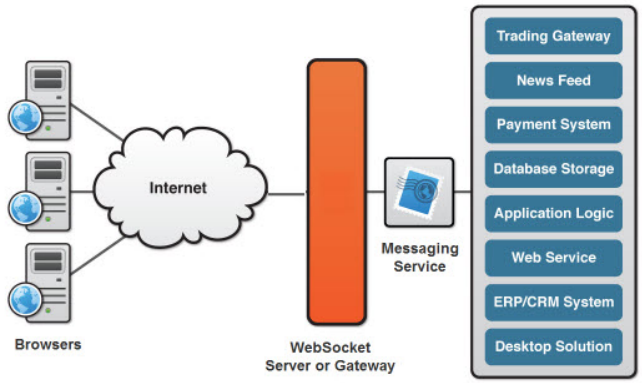


Figura: Architettura WebSocket

Soluzione Proposta — Apache Spark

La libreria Spark

Apache Spark è un sistema di cluster computing di tipo general-purpose, scalabile e veloce. Tra le caratteristiche principali abbiamo:

- API di alto livello in **Java**, **Scala**, **Python** ed **R**
- Un engine ottimizzato che supporta **grafi di esecuzione generici**.
- Multipiattaforma (Windows e sistemi UNIX-like)

Tools

- **Spark Streaming** Libreria per lo stream processing real-time
- **Spark SQL** Libreria per Structured Data Processing in stile SQL
- **Spark MLlib** Suite di algoritmi per il Machine Learning distribuito.

Soluzione Proposta — Apache Spark

La libreria Spark

Apache Spark è un sistema di cluster computing di tipo general-purpose, scalabile e veloce. Tra le caratteristiche principali abbiamo:

- API di alto livello in **Java**, **Scala**, **Python** ed **R**
- Un engine ottimizzato che supporta **grafi di esecuzione generici**.
- Multipiattaforma (Windows e sistemi UNIX-like)

Tools

- **Spark Streaming** Libreria per lo stream processing real-time
- **Spark SQL** Libreria per Structured Data Processing in stile SQL
- **Spark MLlib** Suite di algoritmi per il Machine Learning distribuito.

Soluzione Proposta — Apache Spark

La libreria Spark

Apache Spark è un sistema di cluster computing di tipo general-purpose, scalabile e veloce. Tra le caratteristiche principali abbiamo:

- API di alto livello in **Java**, **Scala**, **Python** ed **R**
- Un engine ottimizzato che supporta **grafi di esecuzione generici**.
- Multipiattaforma (Windows e sistemi UNIX-like)

Tools

- **Spark Streaming** Libreria per lo stream processing real-time
- **Spark SQL** Libreria per Structured Data Processing in stile SQL
- **Spark MLlib** Suite di algoritmi per il Machine Learning distribuito.

Soluzione Proposta — Apache Spark

La libreria Spark

Apache Spark è un sistema di cluster computing di tipo general-purpose, scalabile e veloce. Tra le caratteristiche principali abbiamo:

- API di alto livello in **Java**, **Scala**, **Python** ed **R**
- Un engine ottimizzato che supporta **grafi di esecuzione generici**.
- Multipiattaforma (Windows e sistemi UNIX-like)

Tools

- **Spark Streaming** Libreria per lo stream processing real-time
- **Spark SQL** Libreria per Structured Data Processing in stile SQL
- **Spark MLlib** Suite di algoritmi per il Machine Learning distribuito.

Soluzione Proposta — Apache Spark

La libreria Spark

Apache Spark è un sistema di cluster computing di tipo general-purpose, scalabile e veloce. Tra le caratteristiche principali abbiamo:

- API di alto livello in **Java**, **Scala**, **Python** ed **R**
- Un engine ottimizzato che supporta **grafi di esecuzione generici**.
- Multipiattaforma (Windows e sistemi UNIX-like)

Tools

- **Spark Streaming** Libreria per lo stream processing real-time
- **Spark SQL** Libreria per Structured Data Processing in stile SQL
- **Spark MLlib** Suite di algoritmi per il Machine Learning distribuito.

Soluzione Proposta — Apache Spark

La libreria Spark

Apache Spark è un sistema di cluster computing di tipo general-purpose, scalabile e veloce. Tra le caratteristiche principali abbiamo:

- API di alto livello in **Java**, **Scala**, **Python** ed **R**
- Un engine ottimizzato che supporta **grafi di esecuzione generici**.
- Multipiattaforma (Windows e sistemi UNIX-like)

Tools

- **Spark Streaming** Libreria per lo stream processing real-time
- **Spark SQL** Libreria per Structured Data Processing in stile SQL
- **Spark MLlib** Suite di algoritmi per il Machine Learning distribuito.

Soluzione Proposta — Apache Spark

La libreria Spark

Apache Spark è un sistema di cluster computing di tipo general-purpose, scalabile e veloce. Tra le caratteristiche principali abbiamo:

- API di alto livello in **Java**, **Scala**, **Python** ed **R**
- Un engine ottimizzato che supporta **grafi di esecuzione generici**.
- Multipiattaforma (Windows e sistemi UNIX-like)

Tools

- **Spark Streaming** Libreria per lo stream processing real-time
- **Spark SQL** Libreria per Structured Data Processing in stile SQL
- **Spark MLlib** Suite di algoritmi per il Machine Learning distribuito.

Soluzione Proposta — Apache Spark

La libreria Spark

Apache Spark è un sistema di cluster computing di tipo general-purpose, scalabile e veloce. Tra le caratteristiche principali abbiamo:

- API di alto livello in **Java**, **Scala**, **Python** ed **R**
- Un engine ottimizzato che supporta **grafi di esecuzione generici**.
- Multipiattaforma (Windows e sistemi UNIX-like)

Tools

- **Spark Streaming** Libreria per lo stream processing real-time
- **Spark SQL** Libreria per Structured Data Processing in stile SQL
- **Spark MLlib** Suite di algoritmi per il Machine Learning distribuito.

Soluzione Proposta — Apache Spark

Architettura di Spark

Le applicazioni sono eseguite come set di processi indipendenti sul cluster. Il flusso di esecuzione prevede:

- Un programma sorgente detto **driver** o nodo **Master**
- Un **Cluster Mananager** per l'allocazione delle risorse.
- Gli **executor** o nodi **worker** che eseguono i task

Pro

- Open Source
- Performance elevate (100 volte più veloce di Hadoop)
- Workflow complessi
- Alta tollerabilità ai guasti

Contro

- Condivisione dei dati nativamente non supportata
- Connessione sempre attiva tra master e workers
- Persistenza dei dati in fase di sviluppo.

Soluzione Proposta — Apache Spark

Architettura di Spark

Le applicazioni sono eseguite come set di processi indipendenti sul cluster. Il flusso di esecuzione prevede:

- Un programma sorgente detto **driver** o nodo **Master**
- Un **Cluster Mananager** per l'allocazione delle risorse.
- Gli **executor** o nodi **worker** che eseguono i task

Pro

- Open Source
- Performance elevate (100 volte più veloce di Hadoop)
- Workflow complessi
- Alta tollerabilità ai guasti

Contro

- Condivisione dei dati nativamente non supportata
- Connessione sempre attiva tra master e workers
- Persistenza dei dati in fase di sviluppo.

Soluzione Proposta — Apache Spark

Architettura di Spark

Le applicazioni sono eseguite come set di processi indipendenti sul cluster. Il flusso di esecuzione prevede:

- Un programma sorgente detto **driver** o nodo **Master**
- Un **Cluster Mananager** per l'allocazione delle risorse.
- Gli **executor** o nodi **worker** che eseguono i task

Pro

- Open Source
- Performance elevate (100 volte più veloce di Hadoop)
- Workflow complessi
- Alta tollerabilità ai guasti

Contro

- Condivisione dei dati nativamente non supportata
- Connessione sempre attiva tra master e workers
- Persistenza dei dati in fase di sviluppo.

Soluzione Proposta — Apache Spark

Architettura di Spark

Le applicazioni sono eseguite come set di processi indipendenti sul cluster. Il flusso di esecuzione prevede:

- Un programma sorgente detto **driver** o nodo **Master**
- Un **Cluster Mananager** per l'allocazione delle risorse.
- Gli **executor** o nodi **worker** che eseguono i task

Pro

- Open Source
- Performance elevate (100 volte più veloce di Hadoop)
- Workflow complessi
- Alta tollerabilità ai guasti

Contro

- Condivisione dei dati nativamente non supportata
- Connessione sempre attiva tra master e workers
- Persistenza dei dati in fase di sviluppo.

Soluzione Proposta — Apache Spark

Architettura di Spark

Le applicazioni sono eseguite come set di processi indipendenti sul cluster. Il flusso di esecuzione prevede:

- Un programma sorgente detto **driver** o nodo **Master**
- Un **Cluster Mananager** per l'allocazione delle risorse.
- Gli **executor** o nodi **worker** che eseguono i task

Pro

- Open Source
- Performance elevate (100 volte più veloce di Hadoop)
- Workflow complessi
- Alta tollerabilità ai guasti

Contro

- Condivisione dei dati nativamente non supportata
- Connessione sempre attiva tra master e workers
- Persistenza dei dati in fase di sviluppo.

Soluzione Proposta — Apache Spark

Architettura di Spark

Le applicazioni sono eseguite come set di processi indipendenti sul cluster. Il flusso di esecuzione prevede:

- Un programma sorgente detto **driver** o nodo **Master**
- Un **Cluster Mananager** per l'allocazione delle risorse.
- Gli **executor** o nodi **worker** che eseguono i task

Pro

- Open Source
- Performance elevate (100 volte più veloce di Hadoop)
- Workflow complessi
- Alta tollerabilità ai guasti

Contro

- Condivisione dei dati nativamente non supportata
- Connessione sempre attiva tra master e workers
- Persistenza dei dati in fase di sviluppo.

Soluzione Proposta — Apache Spark

Architettura di Spark

Le applicazioni sono eseguite come set di processi indipendenti sul cluster. Il flusso di esecuzione prevede:

- Un programma sorgente detto **driver** o nodo **Master**
- Un **Cluster Mananager** per l'allocazione delle risorse.
- Gli **executor** o nodi **worker** che eseguono i task

Pro

- Open Source
- Performance elevate (100 volte più veloce di Hadoop)
- Workflow complessi
- Alta tollerabilità ai guasti

Contro

- Condivisione dei dati nativamente non supportata
- Connessione sempre attiva tra master e workers
- Persistenza dei dati in fase di sviluppo.

Soluzione Proposta — Apache Spark

Architettura di Spark

Le applicazioni sono eseguite come set di processi indipendenti sul cluster. Il flusso di esecuzione prevede:

- Un programma sorgente detto **driver** o nodo **Master**
- Un **Cluster Mananager** per l'allocazione delle risorse.
- Gli **executor** o nodi **worker** che eseguono i task

Pro

- Open Source
- Performance elevate (100 volte più veloce di Hadoop)
- Workflow complessi
- Alta tollerabilità ai guasti

Contro

- Condivisione dei dati nativamente non supportata
- Connessione sempre attiva tra master e workers
- Persistenza dei dati in fase di sviluppo.

Soluzione Proposta — Apache Spark

Architettura di Spark

Le applicazioni sono eseguite come set di processi indipendenti sul cluster. Il flusso di esecuzione prevede:

- Un programma sorgente detto **driver** o nodo **Master**
- Un **Cluster Mananager** per l'allocazione delle risorse.
- Gli **executor** o nodi **worker** che eseguono i task

Pro

- Open Source
- Performance elevate (100 volte più veloce di Hadoop)
- Workflow complessi
- Alta tollerabilità ai guasti

Contro

- Condivisione dei dati nativamente non supportata
- Connessione sempre attiva tra master e workers
- Persistenza dei dati in fase di sviluppo.

Soluzione Proposta — Apache Spark

Architettura di Spark

Le applicazioni sono eseguite come set di processi indipendenti sul cluster. Il flusso di esecuzione prevede:

- Un programma sorgente detto **driver** o nodo **Master**
- Un **Cluster Mananager** per l'allocazione delle risorse.
- Gli **executor** o nodi **worker** che eseguono i task

Pro

- Open Source
- Performance elevate (100 volte più veloce di Hadoop)
- Workflow complessi
- Alta tollerabilità ai guasti

Contro

- Condivisione dei dati nativamente non supportata
- Connessione sempre attiva tra master e workers
- Persistenza dei dati in fase di sviluppo.

Soluzione Proposta — Apache Spark

Architettura di Spark

Le applicazioni sono eseguite come set di processi indipendenti sul cluster. Il flusso di esecuzione prevede:

- Un programma sorgente detto **driver** o nodo **Master**
- Un **Cluster Mananager** per l'allocazione delle risorse.
- Gli **executor** o nodi **worker** che eseguono i task

Pro

- Open Source
- Performance elevate (100 volte più veloce di Hadoop)
- Workflow complessi
- Alta tollerabilità ai guasti

Contro

- Condivisione dei dati nativamente non supportata
- Connessione sempre attiva tra master e workers
- Persistenza dei dati in fase di sviluppo.

Soluzione Proposta — Apache Spark

Architettura di Spark

Le applicazioni sono eseguite come set di processi indipendenti sul cluster. Il flusso di esecuzione prevede:

- Un programma sorgente detto **driver** o nodo **Master**
- Un **Cluster Mananager** per l'allocazione delle risorse.
- Gli **executor** o nodi **worker** che eseguono i task

Pro

- Open Source
- Performance elevate (100 volte più veloce di Hadoop)
- Workflow complessi
- Alta tollerabilità ai guasti

Contro

- Condivisione dei dati nativamente non supportata
- Connessione sempre attiva tra master e workers
- Persistenza dei dati in fase di sviluppo.

Soluzione Proposta — Apache Spark

Architettura di Spark

Le applicazioni sono eseguite come set di processi indipendenti sul cluster. Il flusso di esecuzione prevede:

- Un programma sorgente detto **driver** o nodo **Master**
- Un **Cluster Mananager** per l'allocazione delle risorse.
- Gli **executor** o nodi **worker** che eseguono i task

Pro

- Open Source
- Performance elevate (100 volte più veloce di Hadoop)
- Workflow complessi
- Alta tollerabilità ai guasti

Contro

- Condivisione dei dati nativamente non supportata
- Connessione sempre attiva tra master e workers
- Persistenza dei dati in fase di sviluppo.

Soluzione Proposta — Apache Spark

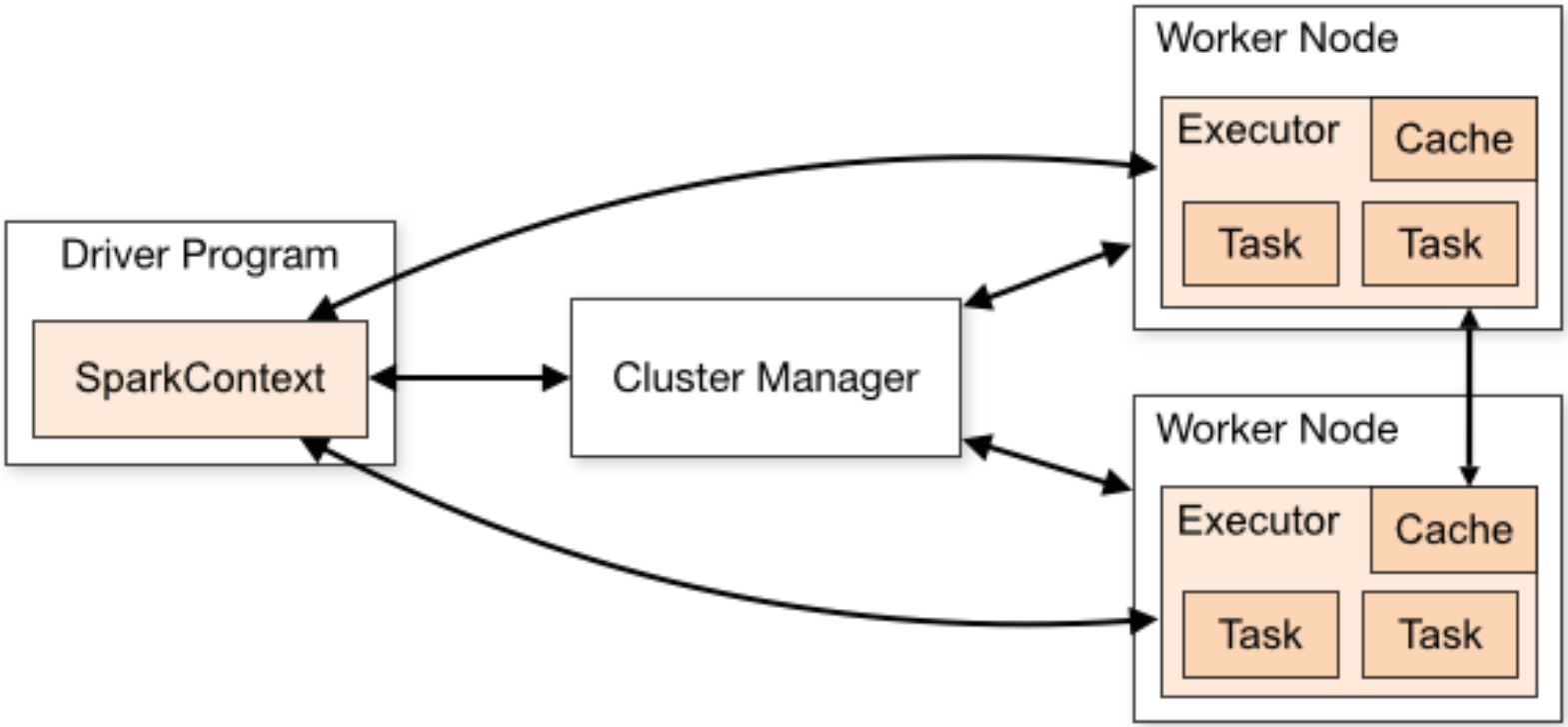


Figura: Architettura Spark

Soluzione Proposta — Apache Spark

Spark Streaming

Estensione delle Core API di Spark per lo **stream processing** di live streams con throughput elevato.

- Supporta differenti sorgenti di stream come **Kafka**, Flume, Twitter,ZeroMQ o socket TCP
- Direttive **map**, **join**, **reduce** e **window**

Architettura

- Lo stream è suddiviso in frammenti detti **batches**
- A livello alto la struttura di riferimento è il **DStream**

Figura: Spark Streaming

Soluzione Proposta — Apache Spark

Spark Streaming

Estensione delle Core API di Spark per lo **stream processing** di live streams con throughput elevato.

- Supporta differenti sorgenti di stream come **Kafka**, Flume, Twitter,ZeroMQ o socket TCP
- Direttive **map**, **join**, **reduce** e **window**

Architettura

- Lo stream è suddiviso in frammenti detti **batches**
- A livello alto la struttura di riferimento è il **DStream**

Figura: Spark Streaming

Soluzione Proposta — Apache Spark

Spark Streaming

Estensione delle Core API di Spark per lo **stream processing** di live streams con throughput elevato.

- Supporta differenti sorgenti di stream come **Kafka**, Flume, Twitter,ZeroMQ o socket TCP
- Direttive **map**, **join**, **reduce** e **window**

Architettura

- Lo stream è suddiviso in frammenti detti **batches**
- A livello alto la struttura di riferimento è il **DStream**

Figura: Spark Streaming

Soluzione Proposta — Apache Spark

Spark Streaming

Estensione delle Core API di Spark per lo **stream processing** di live streams con throughput elevato.

- Supporta differenti sorgenti di stream come **Kafka**, Flume, Twitter,ZeroMQ o socket TCP
- Direttive **map**, **join**, **reduce** e **window**

Architettura

- Lo stream è suddiviso in frammenti detti **batches**
- A livello alto la struttura di riferimento è il **DStream**

Figura: Spark Streaming

Soluzione Proposta — Apache Spark

Spark Streaming

Estensione delle Core API di Spark per lo **stream processing** di live streams con throughput elevato.

- Supporta differenti sorgenti di stream come **Kafka**, Flume, Twitter,ZeroMQ o socket TCP
- Direttive **map**, **join**, **reduce** e **window**

Architettura

- Lo stream è suddiviso in frammenti detti **batches**
- A livello alto la struttura di riferimento è il **DStream**

Figura: Spark Streaming

Soluzione Proposta — Apache Spark

Spark Streaming

Estensione delle Core API di Spark per lo **stream processing** di live streams con throughput elevato.

- Supporta differenti sorgenti di stream come **Kafka**, Flume, Twitter,ZeroMQ o socket TCP
- Direttive **map**, **join**, **reduce** e **window**

Architettura

- Lo stream è suddiviso in frammenti detti **batches**
- A livello alto la struttura di riferimento è il **DStream**

Figura: Spark Streaming

Soluzione Proposta — Apache Spark

Apache Spark SQL

Modulo di Spark per il processing di dati strutturati. Consente di effettuare il processing sui dati sfruttando gli operatori del linguaggio **SQL**. Le strutture di riferimento in Spark sono i **DataFrame** e i **DataSet**.

Apache MLlib

Collezione di algoritmi per il machine learning distribuito. Il suo obiettivo è quello di rendere tali procedure semplici e scalabili. Tra le implementazioni disponibili abbiamo:

- Classificazione
- Regressione
- Clustering
- Collaborative Filtering
- Feature Selection

Soluzione Proposta — Apache Spark

Apache Spark SQL

Modulo di Spark per il processing di dati strutturati. Consente di effettuare il processing sui dati sfruttando gli operatori del linguaggio **SQL**. Le strutture di riferimento in Spark sono i **DataFrame** e i **DataSet**.

Apache MLlib

Collezione di algoritmi per il machine learning distribuito. Il suo obiettivo è quello di rendere tali procedure semplici e scalabili. Tra le implementazioni disponibili abbiamo:

- Classificazione
- Regressione
- Clustering
- Collaborative Filtering
- Feature Selection

Soluzione Proposta — Apache Spark

Apache Spark SQL

Modulo di Spark per il processing di dati strutturati. Consente di effettuare il processing sui dati sfruttando gli operatori del linguaggio **SQL**. Le strutture di riferimento in Spark sono i **DataFrame** e i **DataSet**.

Apache MLlib

Collezione di algoritmi per il machine learning distribuito. Il suo obiettivo è quello di rendere tali procedure semplici e scalabili. Tra le implementazioni disponibili abbiamo:

- Classificazione
- Regressione
- Clustering
- Collaborative Filtering
- Feature Selection

Soluzione Proposta — Apache Spark

Apache Spark SQL

Modulo di Spark per il processing di dati strutturati. Consente di effettuare il processing sui dati sfruttando gli operatori del linguaggio **SQL**. Le strutture di riferimento in Spark sono i **DataFrame** e i **DataSet**.

Apache MLlib

Collezione di algoritmi per il machine learning distribuito. Il suo obiettivo è quello di rendere tali procedure semplici e scalabili. Tra le implementazioni disponibili abbiamo:

- Classificazione
- Regressione
- Clustering
- Collaborative Filtering
- Feature Selection

Soluzione Proposta — Apache Spark

Apache Spark SQL

Modulo di Spark per il processing di dati strutturati. Consente di effettuare il processing sui dati sfruttando gli operatori del linguaggio **SQL**. Le strutture di riferimento in Spark sono i **DataFrame** e i **DataSet**.

Apache MLlib

Collezione di algoritmi per il machine learning distribuito. Il suo obiettivo è quello di rendere tali procedure semplici e scalabili. Tra le implementazioni disponibili abbiamo:

- Classificazione
- Regressione
- Clustering
- Collaborative Filtering
- Feature Selection

Soluzione Proposta — Apache Spark

Apache Spark SQL

Modulo di Spark per il processing di dati strutturati. Consente di effettuare il processing sui dati sfruttando gli operatori del linguaggio **SQL**. Le strutture di riferimento in Spark sono i **DataFrame** e i **DataSet**.

Apache MLlib

Collezione di algoritmi per il machine learning distribuito. Il suo obiettivo è quello di rendere tali procedure semplici e scalabili. Tra le implementazioni disponibili abbiamo:

- Classificazione
- Regressione
- Clustering
- Collaborative Filtering
- Feature Selection

Soluzione Proposta — Apache Spark

Apache Spark SQL

Modulo di Spark per il processing di dati strutturati. Consente di effettuare il processing sui dati sfruttando gli operatori del linguaggio **SQL**. Le strutture di riferimento in Spark sono i **DataFrame** e i **DataSet**.

Apache MLlib

Collezione di algoritmi per il machine learning distribuito. Il suo obiettivo è quello di rendere tali procedure semplici e scalabili. Tra le implementazioni disponibili abbiamo:

- Classificazione
- Regressione
- Clustering
- Collaborative Filtering
- Feature Selection

Soluzione Proposta — Apache Kafka

Apache Kafka

Messaggistica di tipo publish-subscribe orientata alla distribuzione.

- Flussi di stream organizzati in **topics**
- Processi **producer** (scrivono sul topic) e **consumer** (leggono dal topic)
- Uno o più **broker** coordinano i processi

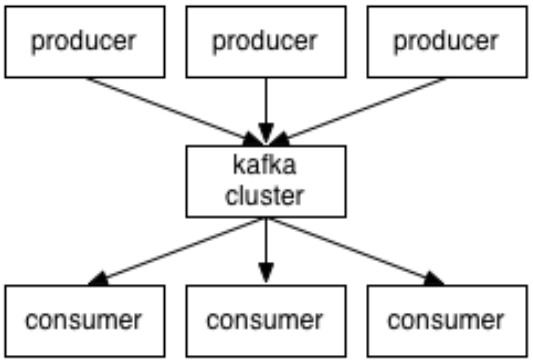


Figura: Architettura Kafka

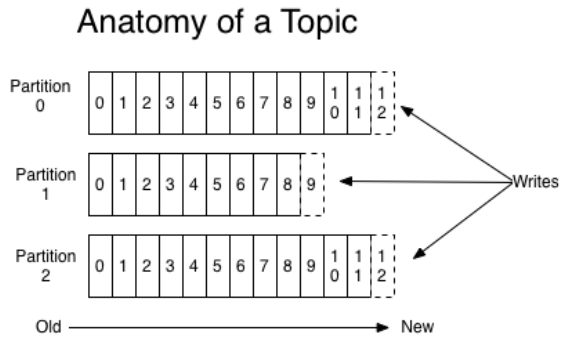


Figura: Struttura di un topic

Soluzione Proposta — Apache Kafka

Apache Kafka

Messaggistica di tipo publish-subscribe orientata alla distribuzione.

- Flussi di stream organizzati in **topics**
- Processi **producer** (scrivono sul topic) e **consumer** (leggono dal topic)
- Uno o più **broker** coordinano i processi

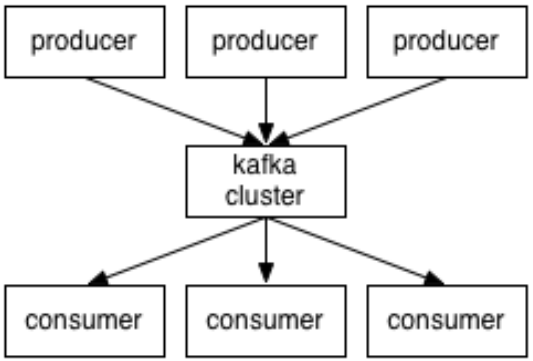


Figura: Architettura Kafka

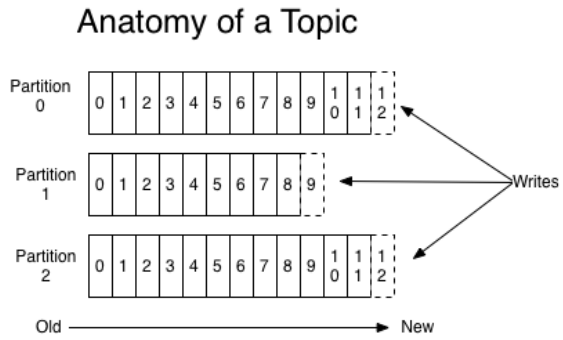


Figura: Struttura di un topic

Soluzione Proposta — Apache Kafka

Apache Kafka

Messaggistica di tipo publish-subscribe orientata alla distribuzione.

- Flussi di stream organizzati in **topics**
- Processi **producer** (scrivono sul topic) e **consumer** (leggono dal topic)
- Uno o più **broker** coordinano i processi

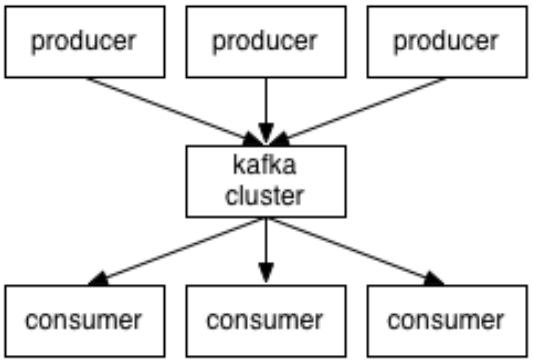


Figura: Architettura Kafka

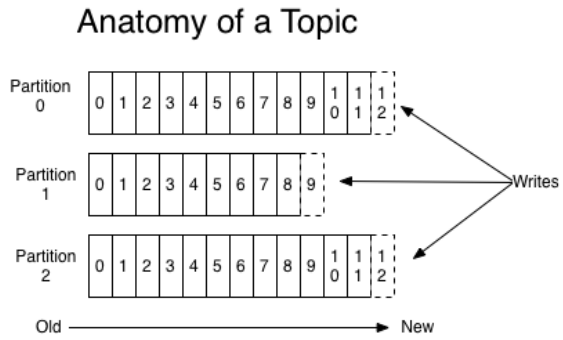


Figura: Struttura di un topic

Soluzione Proposta — Apache Kafka

Apache Kafka

Messaggistica di tipo publish-subscribe orientata alla distribuzione.

- Flussi di stream organizzati in **topics**
- Processi **producer** (scrivono sul topic) e **consumer** (leggono dal topic)
- Uno o più **broker** coordinano i processi

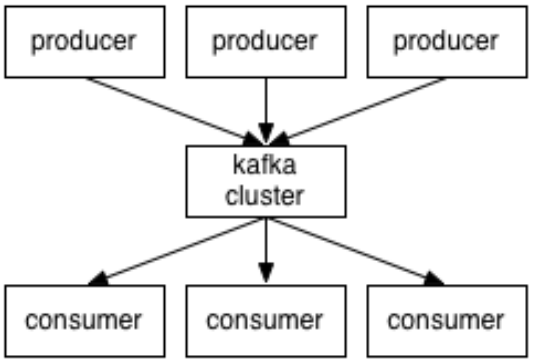


Figura: Architettura Kafka

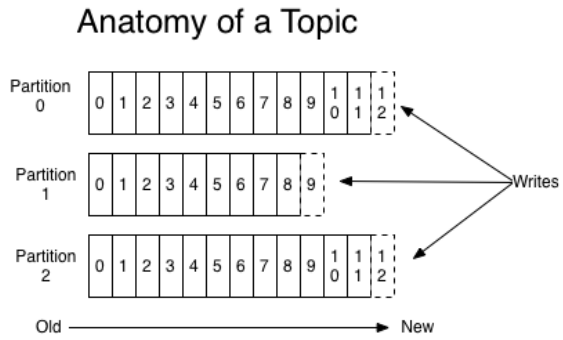


Figura: Struttura di un topic

Soluzione Proposta — Apache Kafka

Apache Kafka

Messaggistica di tipo publish-subscribe orientata alla distribuzione.

- Flussi di stream organizzati in **topics**
- Processi **producer** (scrivono sul topic) e **consumer** (leggono dal topic)
- Uno o più **broker** coordinano i processi

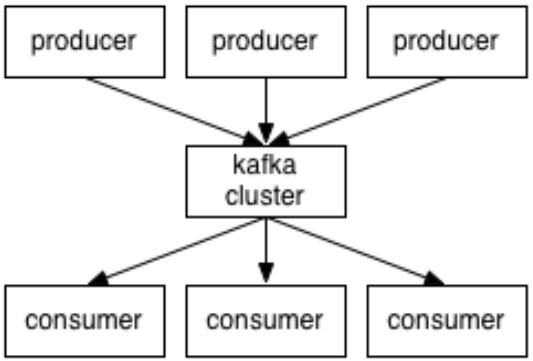


Figura: Architettura Kafka

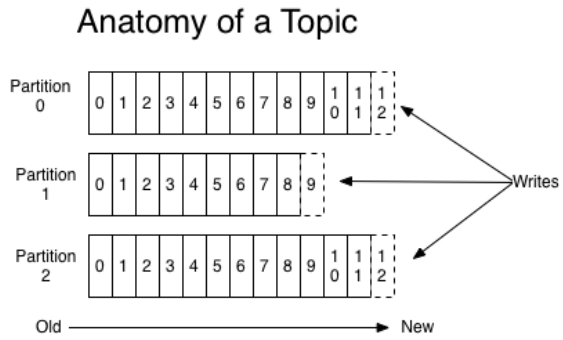


Figura: Struttura di un topic

Soluzione Proposta — Apache Kafka

Apache Kafka

Messaggistica di tipo publish-subscribe orientata alla distribuzione.

- Flussi di stream organizzati in **topics**
- Processi **producer** (scrivono sul topic) e **consumer** (leggono dal topic)
- Uno o più **broker** coordinano i processi

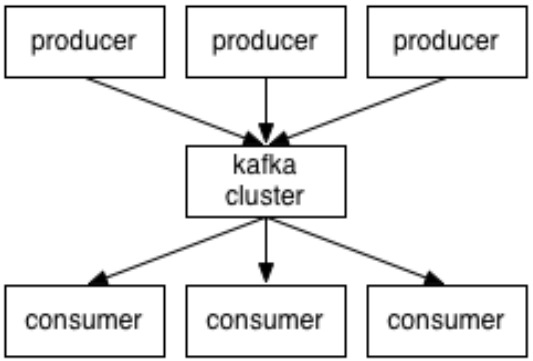


Figura: Architettura Kafka

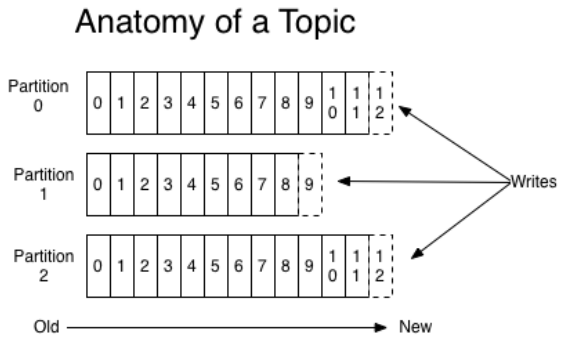


Figura: Struttura di un topic

Soluzione Proposta — Apache Kafka

Apache Kafka

Messaggistica di tipo publish-subscribe orientata alla distribuzione.

- Flussi di stream organizzati in **topics**
- Processi **producer** (scrivono sul topic) e **consumer** (leggono dal topic)
- Uno o più **broker** coordinano i processi

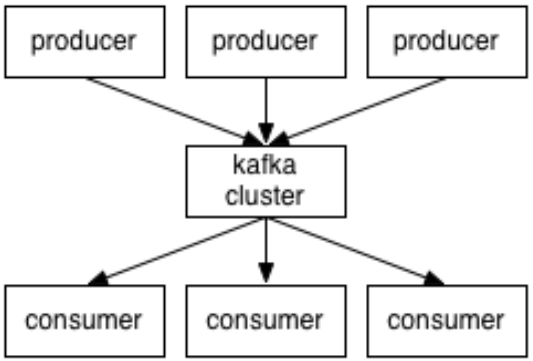


Figura: Architettura Kafka

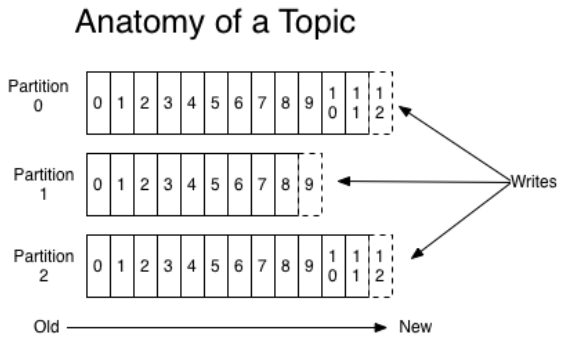
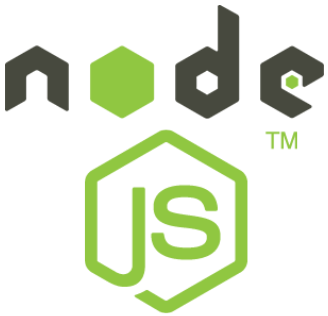


Figura: Struttura di un topic

Soluzione Proposta — Node.js + Angular.js

Node.js



Applicazioni web in **Javascript** lato server. Si basa sul **Javascript Engine V8** di Google.

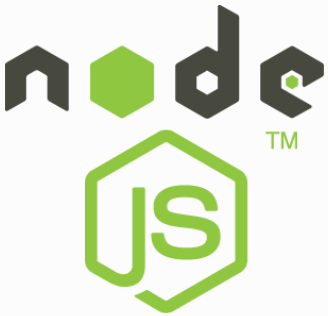
- Modello di esecuzione event-driven
- Scalabile e flessibile
- Package manager npm

Angular.js

AngularJS è un framework JavaScript per lo sviluppo di applicazioni Web client side. Permette di definire interfacce grafiche tramite l'approccio dichiarativo dell'**HTML** esaltandone le potenzialità.

Soluzione Proposta — Node.js + Angular.js

Node.js



Applicazioni web in **Javascript** lato server. Si basa sul **Javascript Engine V8** di Google.

- Modello di esecuzione **event-driven**
- Scalabile e flessibile
- Package manager **npm**

Angular.js

AngularJS è un framework JavaScript per lo sviluppo di applicazioni Web client side. Permette di definire interfacce grafiche tramite l'approccio dichiarativo dell'**HTML** esaltandone le potenzialità.

Soluzione Proposta — Node.js + Angular.js

Node.js

Applicazioni web in **Javascript** lato server. Si basa sul **Javascript Engine V8** di Google.

- Modello di esecuzione **event-driven**
- Scalabile e flessibile
- Package manager **npm**

Angular.js

AngularJS è un framework JavaScript per lo sviluppo di applicazioni Web client side. Permette di definire interfacce grafiche tramite l'approccio dichiarativo dell'**HTML** esaltandone le potenzialità.

Soluzione Proposta — Node.js + Angular.js

Node.js

Applicazioni web in **Javascript** lato server. Si basa sul **Javascript Engine V8** di Google.

- Modello di esecuzione **event-driven**
- Scalabile e flessibile
- Package manager **npm**

Angular.js

AngularJS è un framework JavaScript per lo sviluppo di applicazioni Web client side. Permette di definire interfacce grafiche tramite l'approccio dichiarativo dell'**HTML** esaltandone le potenzialità.

Soluzione Proposta — Node.js + Angular.js

Node.js

Applicazioni web in **Javascript** lato server. Si basa sul **Javascript Engine V8** di Google.

- Modello di esecuzione **event-driven**
- Scalabile e flessibile
- Package manager **npm**

Angular.js

AngularJS è un framework JavaScript per lo sviluppo di applicazioni Web client side. Permette di definire interfacce grafiche tramite l'approccio dichiarativo dell'**HTML** esaltandone le potenzialità.

Soluzione Proposta — Node.js + Angular.js

Node.js

Applicazioni web in **Javascript** lato server. Si basa sul **Javascript Engine V8** di Google.

- Modello di esecuzione **event-driven**
- Scalabile e flessibile
- Package manager **npm**

Angular.js

AngularJS è un framework JavaScript per lo sviluppo di applicazioni Web client side. Permette di definire interfacce grafiche tramite l'approccio dichiarativo dell'**HTML** esaltandone le potenzialità.

Soluzione Proposta — Node.js + Angular.js

Node.js

Applicazioni web in **Javascript** lato server. Si basa sul **Javascript Engine V8** di Google.

- Modello di esecuzione **event-driven**
- Scalabile e flessibile
- Package manager **npm**

Angular.js

AngularJS è un framework JavaScript per lo sviluppo di applicazioni Web client side. Permette di definire interfacce grafiche tramite l'approccio dichiarativo dell'**HTML** esaltandone le potenzialità.

- Pattern MVC
- Binding bidirezionale (two-way binding)
- Dependency Injection

Soluzione Proposta — Node.js + Angular.js

Node.js

Applicazioni web in **Javascript** lato server. Si basa sul **Javascript Engine V8** di Google.

- Modello di esecuzione **event-driven**
- Scalabile e flessibile
- Package manager **npm**

Angular.js

AngularJS è un framework JavaScript per lo sviluppo di applicazioni Web client side. Permette di definire interfacce grafiche tramite l'approccio dichiarativo dell'**HTML** esaltandone le potenzialità.

- Pattern MVC
- Binding bidirezionale (two-way binding)
- Dependency Injection

Soluzione Proposta — Node.js + Angular.js

Node.js

Applicazioni web in **Javascript** lato server. Si basa sul **Javascript Engine V8** di Google.

- Modello di esecuzione **event-driven**
- Scalabile e flessibile
- Package manager **npm**

Angular.js

AngularJS è un framework JavaScript per lo sviluppo di applicazioni Web client side. Permette di definire interfacce grafiche tramite l'approccio dichiarativo dell'**HTML** esaltandone le potenzialità.

- Pattern MVC
- Binding bidirezionale (two-way binding)
- Dependency Injection

Soluzione Proposta — Node.js + Angular.js

Node.js

Applicazioni web in **Javascript** lato server. Si basa sul **Javascript Engine V8** di Google.

- Modello di esecuzione **event-driven**
- Scalabile e flessibile
- Package manager **npm**

Angular.js

AngularJS è un framework JavaScript per lo sviluppo di applicazioni Web client side. Permette di definire interfacce grafiche tramite l'approccio dichiarativo dell'**HTML** esaltandone le potenzialità.

- Pattern MVC
- Binding bidirezionale (two-way binding)
- Dependency Injection

Soluzione Proposta — Node.js + Angular.js

Node.js

Applicazioni web in **Javascript** lato server. Si basa sul **Javascript Engine V8** di Google.

- Modello di esecuzione **event-driven**
- Scalabile e flessibile
- Package manager **npm**

Angular.js

AngularJS è un framework JavaScript per lo sviluppo di applicazioni Web client side. Permette di definire interfacce grafiche tramite l'approccio dichiarativo dell'**HTML** esaltandone le potenzialità.

- Pattern MVC
- Binding bidirezionale (two-way binding)
- Dependency Injection

Analisi e progettazione della soluzione proposta

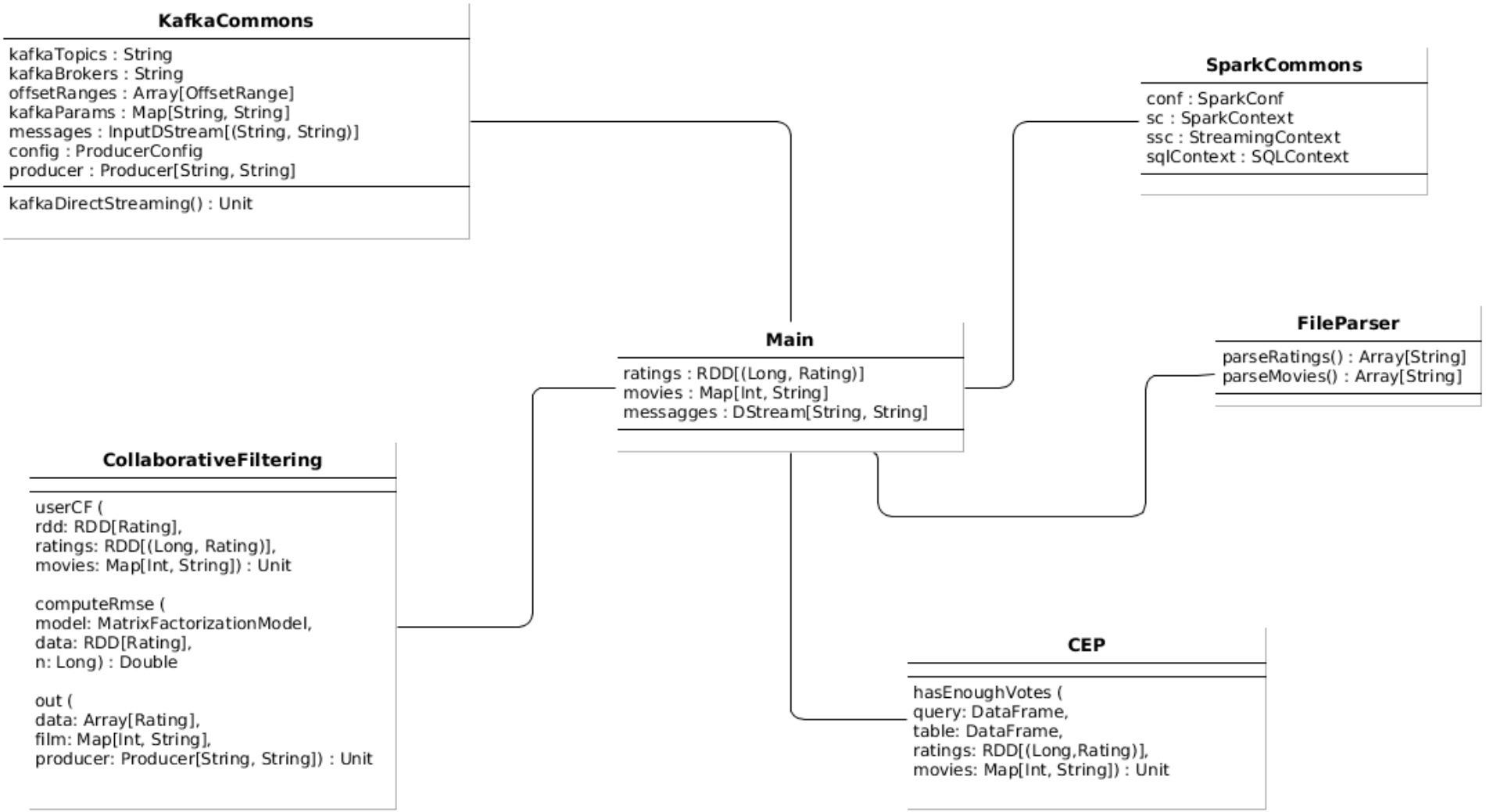


Figura: Classi dell'applicazione

Classi dell'applicazione

Classi

- **SparkCommons**: Si occupa di inizializzare a livello globale il driver di Spark.
- **KafkaCommons**: Configura l'interfaccia verso il server Kafka, definendo la lista dei broker, dei topic ed inizializzando il Producer per la notifica delle raccomandazioni effettuate.
- **CEP**: La classe implementa il metodo *hasEnoughVotes*, il quale effettua il pattern matching sul DataFrame dall'aggregazione dei voti ricevuti nella finestra temporale.
- **CollaborativeFiltering**: Rappresenta il cuore del sistema. Esegue l'algoritmo ALS sul set di dati ricevuti. Si occupa della predizione degli n rating e del calcolo dell'RMSE sul validation set. Serializzazione dell'output in formato JSON ed invio dei dati sul topic creato dal server Kafka.
- **FileParser**: I metodi parseRatings e parseMovies, si occupano di parsare i voti e i film presenti nel dataset.
- **Main**: Interfaccia dell'intera applicazione, che consente di avviare avviando la pipeline di operazioni messe a disposizione dagli oggetti precedentemente descritti.

Classi dell'applicazione

Classi

- **SparkCommons**: Si occupa di inizializzare a livello globale il driver di Spark.
- **KafkaCommons**: Configura l'interfaccia verso il server Kafka, definendo la lista dei broker, dei topic ed inizializzando il Producer per la notifica delle raccomandazioni effettuate.
- **CEP**: La classe implementa il metodo *hasEnoughVotes*, il quale effettua il pattern matching sul DataFrame dall'aggregazione dei voti ricevuti nella finestra temporale.
- **CollaborativeFiltering**: Rappresenta il cuore del sistema. Esegue l'algoritmo ALS sul set di dati ricevuti. Si occupa della predizione degli n rating e del calcolo dell'RMSE sul validation set. Serializzazione dell'output in formato JSON ed invio dei dati sul topic creato dal server Kafka.
- **FileParser**: I metodi parseRatings e parseMovies, si occupano di parsare i voti e i film presenti nel dataset.
- **Main**: Interfaccia dell'intera applicazione, che consente di avviare avviando la pipeline di operazioni messe a disposizione dagli oggetti precedentemente descritti.

Classi dell'applicazione

Classi

- **SparkCommons**: Si occupa di inizializzare a livello globale il driver di Spark.
- **KafkaCommons**: Configura l'interfaccia verso il server Kafka, definendo la lista dei broker, dei topic ed inizializzando il Producer per la notifica delle raccomandazioni effettuate.
- **CEP**: La classe implementa il metodo *hasEnoughVotes*, il quale effettua il pattern matching sul DataFrame dall'aggregazione dei voti ricevuti nella finestra temporale.
- **CollaborativeFiltering**: Rappresenta il cuore del sistema. Esegue l'algoritmo ALS sul set di dati ricevuti. Si occupa della predizione degli n rating e del calcolo dell'RMSE sul validation set. Serializzazione dell'output in formato JSON ed invio dei dati sul topic creato dal server Kafka.
- **FileParser**: I metodi parseRatings e parseMovies, si occupano di parsare i voti e i film presenti nel dataset.
- **Main**: Interfaccia dell'intera applicazione, che consente di avviare avviando la pipeline di operazioni messe a disposizione dagli oggetti precedentemente descritti.

Classi dell'applicazione

Classi

- **SparkCommons**: Si occupa di inizializzare a livello globale il driver di Spark.
- **KafkaCommons**: Configura l'interfaccia verso il server Kafka, definendo la lista dei broker, dei topic ed inizializzando il Producer per la notifica delle raccomandazioni effettuate.
- **CEP**: La classe implementa il metodo *hasEnoughVotes*, il quale effettua il pattern matching sul DataFrame dall'aggregazione dei voti ricevuti nella finestra temporale.
- **CollaborativeFiltering**: Rappresenta il cuore del sistema. Esegue l'algoritmo ALS sul set di dati ricevuti. Si occupa della predizione degli *n* rating e del calcolo dell'RMSE sul validation set. Serializzazione dell'output in formato JSON ed invio dei dati sul topic creato dal server Kafka.
- **FileParser**: I metodi parseRatings e parseMovies, si occupano di parsare i voti e i film presenti nel dataset.
- **Main**: Interfaccia dell'intera applicazione, che consente di avviare avviando la pipeline di operazioni messe a disposizione dagli oggetti precedentemente descritti.

Classi dell'applicazione

Classi

- **SparkCommons**: Si occupa di inizializzare a livello globale il driver di Spark.
- **KafkaCommons**: Configura l'interfaccia verso il server Kafka, definendo la lista dei broker, dei topic ed inizializzando il Producer per la notifica delle raccomandazioni effettuate.
- **CEP**: La classe implementa il metodo *hasEnoughVotes*, il quale effettua il pattern matching sul DataFrame dall'aggregazione dei voti ricevuti nella finestra temporale.
- **CollaborativeFiltering**: Rappresenta il cuore del sistema. Esegue l'algoritmo ALS sul set di dati ricevuti. Si occupa della predizione degli *n* rating e del calcolo dell'RMSE sul validation set. Serializzazione dell'output in formato JSON ed invio dei dati sul topic creato dal server Kafka.
- **FileParser**: I metodi parseRatings e parseMovies, si occupano di parsare i voti e i film presenti nel dataset.
- **Main**: Interfaccia dell'intera applicazione, che consente di avviare avviando la pipeline di operazioni messe a disposizione dagli oggetti precedentemente descritti.

Classi dell'applicazione

Classi

- **SparkCommons**: Si occupa di inizializzare a livello globale il driver di Spark.
- **KafkaCommons**: Configura l'interfaccia verso il server Kafka, definendo la lista dei broker, dei topic ed inizializzando il Producer per la notifica delle raccomandazioni effettuate.
- **CEP**: La classe implementa il metodo *hasEnoughVotes*, il quale effettua il pattern matching sul DataFrame dall'aggregazione dei voti ricevuti nella finestra temporale.
- **CollaborativeFiltering**: Rappresenta il cuore del sistema. Esegue l'algoritmo ALS sul set di dati ricevuti. Si occupa della predizione degli *n* rating e del calcolo dell'RMSE sul validation set. Serializzazione dell'output in formato JSON ed invio dei dati sul topic creato dal server Kafka.
- **FileParser**: I metodi parseRatings e parseMovies, si occupano di parsare i voti e i film presenti nel dataset.
- **Main**: Interfaccia dell'intera applicazione, che consente di avviare avviando la pipeline di operazioni messe a disposizione dagli oggetti precedentemente descritti.

Classi dell'applicazione

Classi

- **SparkCommons**: Si occupa di inizializzare a livello globale il driver di Spark.
- **KafkaCommons**: Configura l'interfaccia verso il server Kafka, definendo la lista dei broker, dei topic ed inizializzando il Producer per la notifica delle raccomandazioni effettuate.
- **CEP**: La classe implementa il metodo *hasEnoughVotes*, il quale effettua il pattern matching sul DataFrame dall'aggregazione dei voti ricevuti nella finestra temporale.
- **CollaborativeFiltering**: Rappresenta il cuore del sistema. Esegue l'algoritmo ALS sul set di dati ricevuti. Si occupa della predizione degli *n* rating e del calcolo dell'RMSE sul validation set. Serializzazione dell'output in formato JSON ed invio dei dati sul topic creato dal server Kafka.
- **FileParser**: I metodi parseRatings e parseMovies, si occupano di parsare i voti e i film presenti nel dataset.
- **Main**: Interfaccia dell'intera applicazione, che consente di avviare avviando la pipeline di operazioni messe a disposizione dagli oggetti precedentemente descritti.

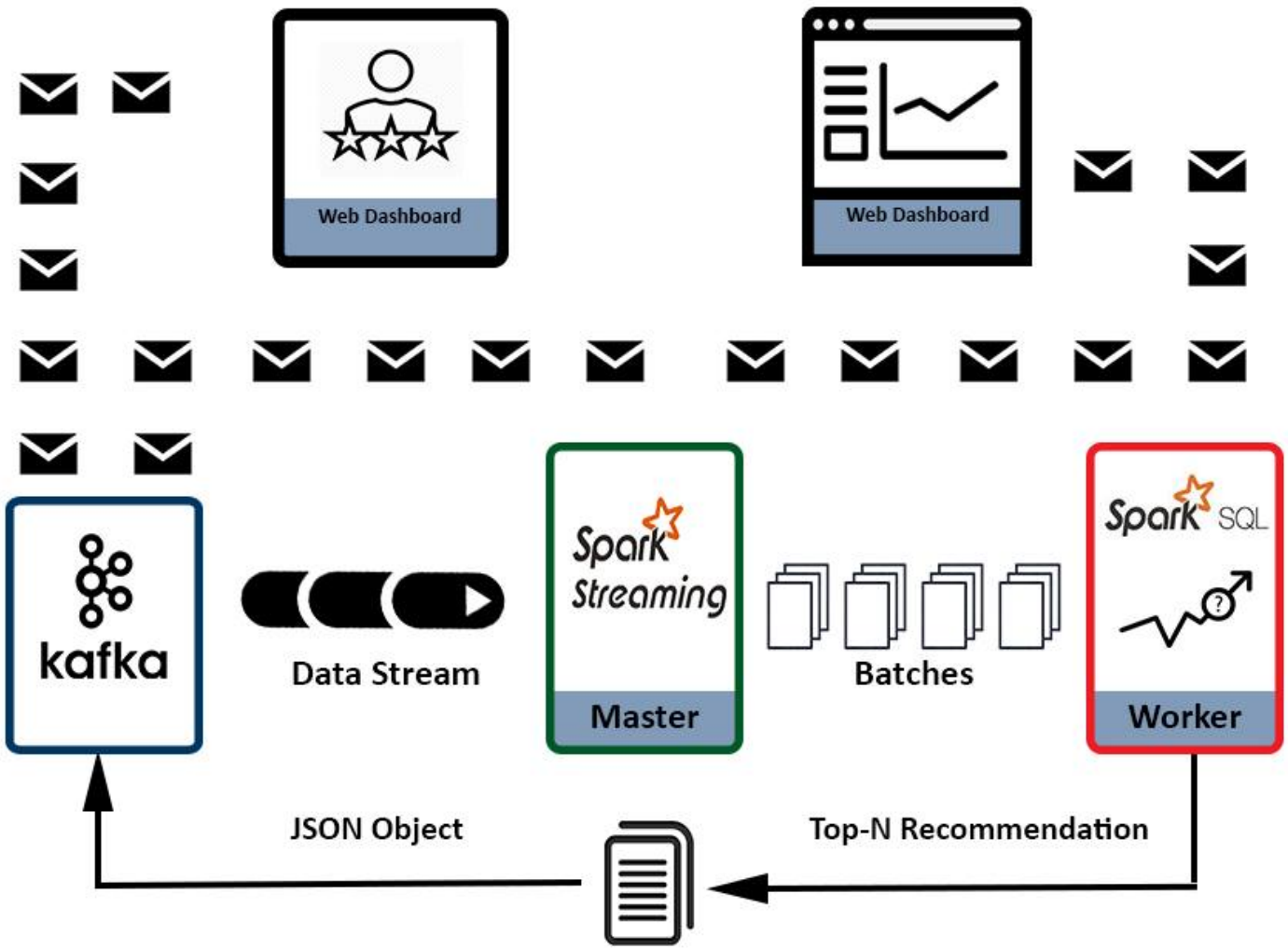


Figura: Architettura del sistema

Conclusioni e Sviluppi Futuri

Conclusioni

Nel presente tema d'anno è stato realizzato un sistema di raccomandazione distribuito per film, in grado di attuare un filtro collaborativo sulla base di uno stream processing e funzionalità di tipo CEP legate al monitoring di tale flusso di dati, il tutto accessibile tramite interfaccia web.

Sviluppi Futuri

Gli sviluppi futuri di tale progetto potrebbero riguardare la **valutazione delle performance**, la scelta di calcolatori più efficienti ed una possibile migrazione su piattaforma **cloud**.

Riferimenti bibliografici I

I. Fette and A. Melnikov, "The websocket protocol," 2011.

B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems," in *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ser. PODS '02. New York, NY, USA: ACM, 2002, pp. 1–16. [Online]. Available: http://doi.acm.org/10.1145/543613.543615

G. Takács, I. Pilászy, B. Németh, and D. Tikk, "Matrix factorization and neighbor based algorithms for the netflix prize problem," in *Proceedings of the 2008 ACM Conference on Recommender Systems*, ser. RecSys '08. New York, NY, USA: ACM, 2008, pp. 267–274. [Online]. Available: http://doi.acm.org/10.1145/1454008.1454049

X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Adv. in Artif. Intell.*, vol. 2009, pp. 4:2–4:2, Jan. 2009. [Online]. Available: http://dx.doi.org/10.1155/2009/421425

G. Cugola and A. Margara, "Processing flows of information: From data stream to complex event processing," *ACM Comput. Surv.*, vol. 44, no. 3, pp. 15:1–15:62, Jun. 2012. [Online]. Available: http://doi.acm.org/10.1145/2187671.2187677

L. J. Fülöp, G. Tóth, R. Rácz, J. Pánczél, T. Gergely, A. Beszédes, and L. Farkas, "Survey on complex event processing and predictive analytics," *Nokia Siemens Networks*, 2010.

Riferimenti bibliografici II

P. G. P. Gantet. Complex event processing. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/3/39/Complex_Event_Processing.jpg

J. Bobadilla, F. Ortega, A. Hernando, and A. GutiéRrez, "Recommender systems survey," *Know.-Based Syst.*, vol. 46, pp. 109–132, Jul. 2013. [Online]. Available: http://dx.doi.org/10.1016/j.knosys.2013.03.012

A. S. Foundation, "Spark Official 1.6.1 Documentation," http://spark.apache.org/docs/latest/, 2016.

——, "Kafka Official 0.9.0 Documentation," http://kafka.apache.org/documentation.html, 2016.

L. Foundation, "Node.js Official Documentation," https://nodejs.org/en/docs/, 2016.

Google, "Angular.js Official Documentation," https://docs.angularjs.org/guide, 2016.