Destinatari

Prof. Tullio Vardanega
Prof. Riccardo Cardin

Redattori

Guglielmo Barison
Davide Donanzan
Pietro Busato

Verificatori

Oscar Konieczny
Veronica Tecchiati

Valutazione dell'utilizzo di Flink

nan1fyteam.unipd@gmail.com

Registro delle Modifiche

Versione	Data	Descrizione	Redattori	Verificatori
1.0.0	2024-08-15	**Approvazione per PB**		
0.1.0	2024-08-15	Aggiunta sezione su PyFlink e correzione errori.	Davide Donanzan, Pietro Busato	Oscar Konieczny, Veronica Tecchiati
0.0.0	2024-08-09	Prima stesura del file.	Davide Donanzan, Pietro Busato	Oscar Konieczny, Veronica Tecchiati

Tabella 1: Registro delle modifiche.

Indice

1 Introduzione

Questo documento viene scritto con il fine di individuare le circostanze e le motivazioni che si celano dietro e hanno portato alla scelta, da parte del team, di utilizzare come tecnologia di stream processing Apache Flink.
"Valutazione dell'utilizzo di Flink" è, quindi, a puro scopo descrittivo e narrativo, e come tale si riserva di essere letto.

2 Stream processing

L'implementazione all'interno del progetto di Apache Flink deriva dalla comprensione e necessità, da parte del team, di dover, in talune istanze, elaborare lo stream grezzo di dati ricevuto da Apache Kafka[G] tramite il mocking dei sensori di raccolta.
I principali sentori di questa necessità e utilità sono emersi in due momenti distinti:

- Durante la creazione del sensore[G] di parcheggio e di pagamento dello stesso;
- Durante la realizzazione, sulla piattaforma Grafana[G], dei widget[G] di Heat Index (temperatura percepita).

Nel primo caso, il team si è ritrovato a dover collegare due stream di dati indipendenti dal punto di vista della raccolta (e quindi non comunicanti), ma strettamente legati l'uno con l'altro, sia per la loro mera esistenza (non esiste un pagamento senza un parcheggio occupato), sia che per la coerenza dei dati raccolti (uno stallo occupato da una macchina per una certa ora non può essere collegato a diversi pagamenti più corti nello stesso lasso di tempo).
Nel secondo caso, invece, era chiaro come risultasse concettualmente sbagliato e in parte inefficiente ricavare alcuni tipi di dati, come quelli dell'Heat Index, lavorando direttamente quelli "sporchi", ricavati dagli stream di Temperature (Temperatura) e Humidity (Umidità) tramite query sul database[G] OLAP[G] Clickhouse[G]. Per quanto esso mostri vantaggi non ignorabili nell'analisi di dati in real-time[G], per tali circostanze, interessate dal progetto, l'utilizzo dello stream processing è più indicato e più efficiente sul lungo termine.
Dopo doverose riflessioni dunque, il team ha deciso di comune accordo di implementare lo stream processing; sebbene esso non fosse stato un requisito richiesto espressamente dalla Proponente[G] alla presentazione del Capitolato[G], l'azienda si è rivelata però interessata a tale opportunità, suggerendo in particolare l'utilizzo della tecnologia Apache Flink.

3 Panoramica sulle caratteristiche di Apache Flink

3.1 Cattarestiche favorevoli

- Appoggiato dalla Proponente;
- Supporta in particolar modo gli stream di dati event-driven;
- Gestisce facilmente l'ingresso di grandi quantità di dati;
- Facilmente scalabile e resistente ai failure.

3.2 Cattarestiche contrarie

- Nuova tecnologia da studiare, imparare e implementare;
- Previsto utilizzo di Java, non molto conosciuto dal team;
- Grado di complessità elevato, amplificato dalla scarsa conoscenza precedentemente descritta;
- Documentazione di supporto non completamente esaustiva e a volte insufficiente.

4 Implementazione di Flink

È doveroso precisare, prima di iniziare la descrizione dell'implementazione, che quest'ultima è stata conseguita utilizzando il linguaggio Java e non tramite l'alternativa per Python[G] "PyFlink", sebbene questa scelta abbia comportato un aumento di complessità rispetto allo stadio precedente del progetto. Si possono di seguito individuare i motivi principali che hanno portato a questa decisione:

- PyFlink presenta una documentazione molto meno estesa rispetto alla controparte di Java, e gode di minor supporto esterno (forum, issue et similia);
- Rispetto alla versione in Java, PyFlink non è supportato allo stesso modo, risultando a tutti gli effetti in una versione peggiore, in talune circostanze obsoleta, che riceve aggiornamenti a posteriori rispetto alla controparte;
- Risulta inoltre più complessa l'implementazione effettiva e il debugging della versione in Python siccome PyFlink è, a tutti gli effetti, un wrapper della versione in Java, facendo comunicare tra di loro gli elementi dei due linguaggi; ne risulta un ovvio aumento di complessità e l'aggiunta di un nuovo e completo dominio di problemi ed errori da affrontare.

Apache Flink è stato implementato attraverso l'utilizzo di due Job (ovvero due "elaborazioni" di stream di dati) differenti, quali:

- HeatIndexJob;
- ParkingEfficiencyJob.

Per il primo Job si è sostanzialmente ricreata la query inizialmente generata su Grafana, che combinava i dati dei Kafka topic[G] Temperature e Humidity attraverso la formula della temperatura percepita: entrambi i topic sono stati utilizzati come Kafka Sources, immettendo i dati all'interno di Flink, elaborandoli e poi scrivendoli nel nuovo topic "Heat Index". Essi vengono poi ricevuti da Clickhouse e successivamente riportati sulle dovute dashboard di Grafana.
Per il secondo Job, invece, come Sources vengono utilizzati i topic "Parking" e "Parking Payment", e il Job si occupa di calcolare l'efficienza monetaria del parcheggio, ricavata tramite la formula Total_Revenue / (Total_Arrivals * Average_Price). Dopodichè tali dati vengono ricevuti da Clickhouse e successivamente riportati all'interno degli appositi pannelli di Grafana.

4.1 Conclusioni

Alla luce delle cause, circostanze e discussioni rilevate finora, nonchè valutate le caratteristiche e ponderato il suggerimento proposto da parte della Proponente[G], il team ha optato per la tecnologia Apache Flink per l'implementazione dello stream processing all'interno del progetto "SyncCity: Smart City Monitoring Platform", secondo le modalità descritte alla sezione 4.