SISTEMI E ARCHITETTURE PER BIG DATA
PROGETTO #2

Stream processing distribuito di Big Data

Con Apache Flink e Apache Kafka

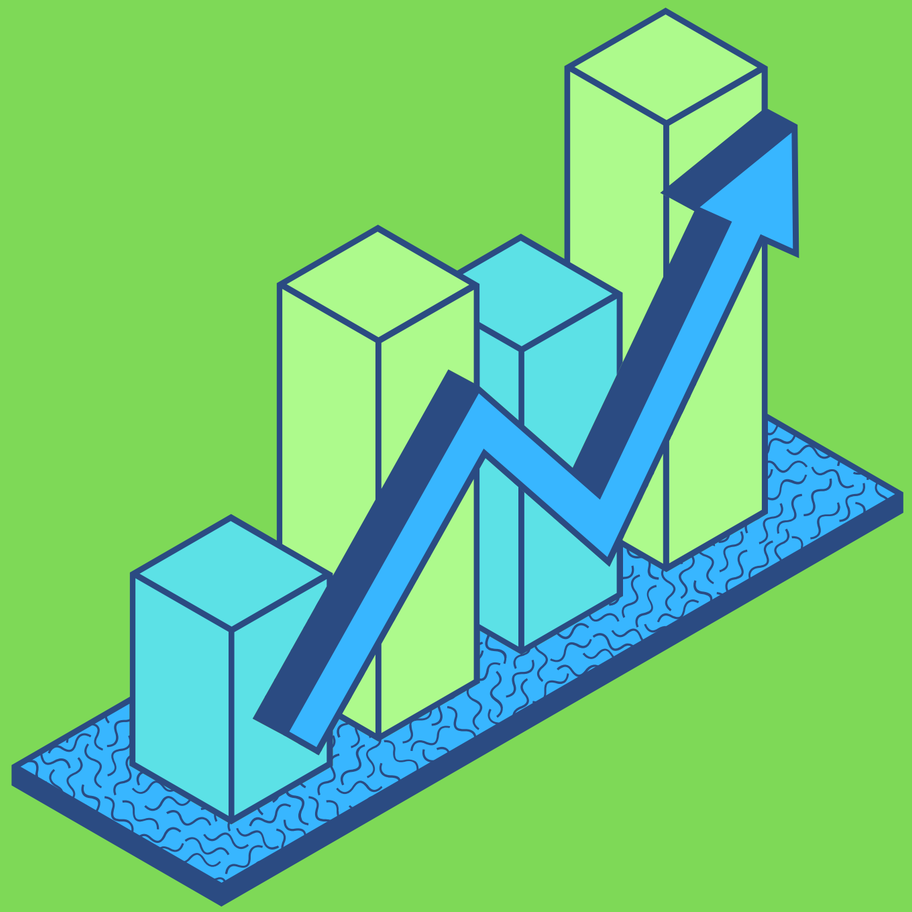
Lo scopo del progetto

Cosa

Analisi quantitativa sul traffico marittimo nel Mar Mediterraneo

Come

Tramite un'architettura di **stream-processing** sviluppata *ad-hoc* con Apache Flink e Apache Kafka



Architettura

1

JAVA APP

Coordinator

Kafka Producer/Consumer inglobato in un singolo e Topology submitter

2

KAFKA

Messaging layer

Trasporto eventi di input e dati di output tramite patter Publish/Subscribe

3

FLINK

Stream processing

Framework di stream-processing per lo sviluppo di applicativi con programmazione Funzionale in Session Mode

4

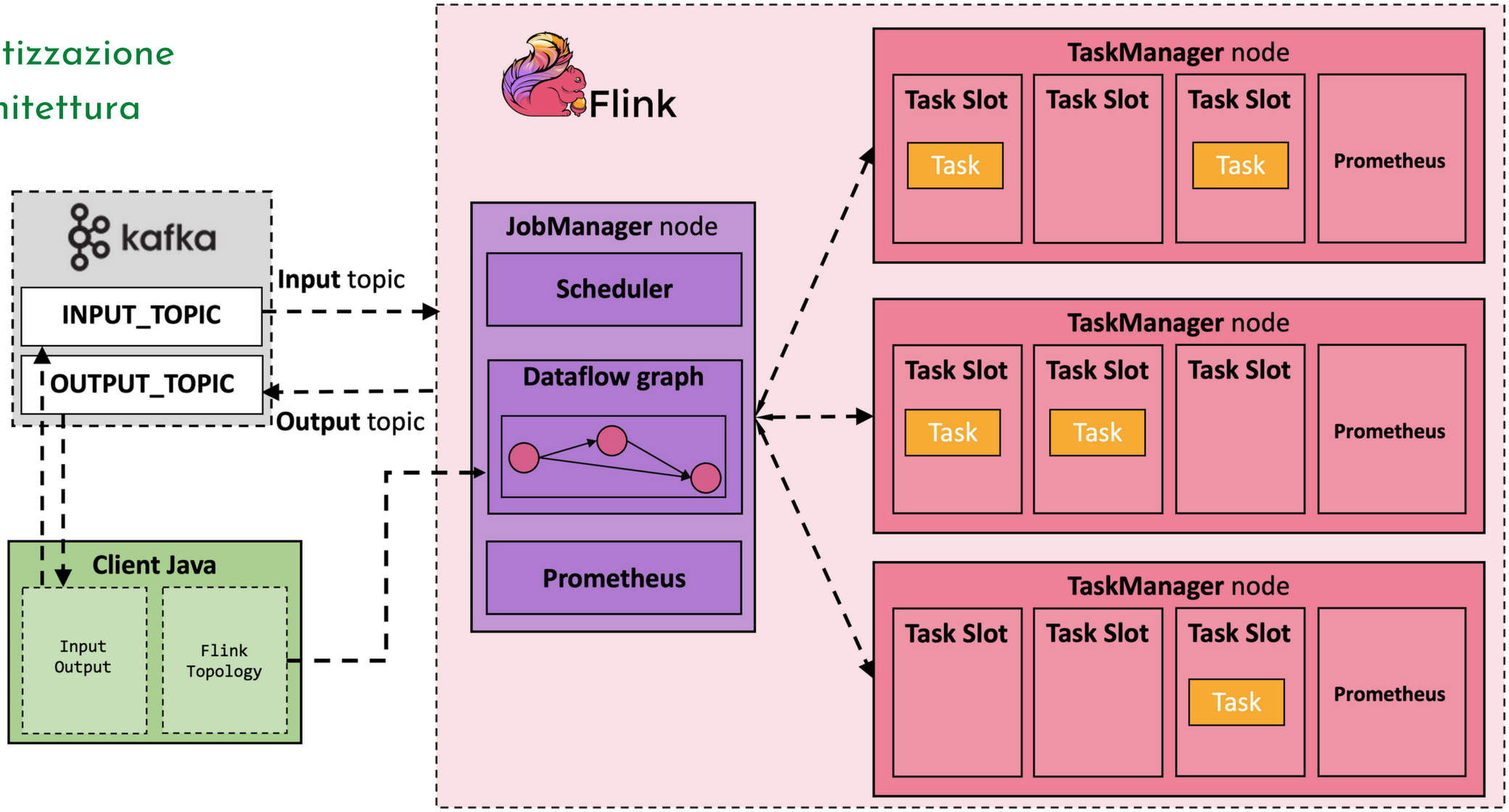
GRAFANA

Monitoring

Tool per la visualizzazione in *real-time* di metriche

As a whole

Schematizzazione
dell'architettura



ARCHITETTURA

- Architettura decentralizzata
- 2 Broker
- 2 Partizioni per Topic
- Grado di replicazione 2
- Topics:
  - INPUT_TOPIC
  - OUTPUT_TOPIC
- ZooKeeper per sincronizzazione e coordinazione

IMPIEGATO PER

- Disaccoppiamento layer di **produzione dati** e layer di **processamento**
- Gestione flusso dati in entrata e in uscita
- **Evento**: Timestamp + Messaggio

Apache Flink

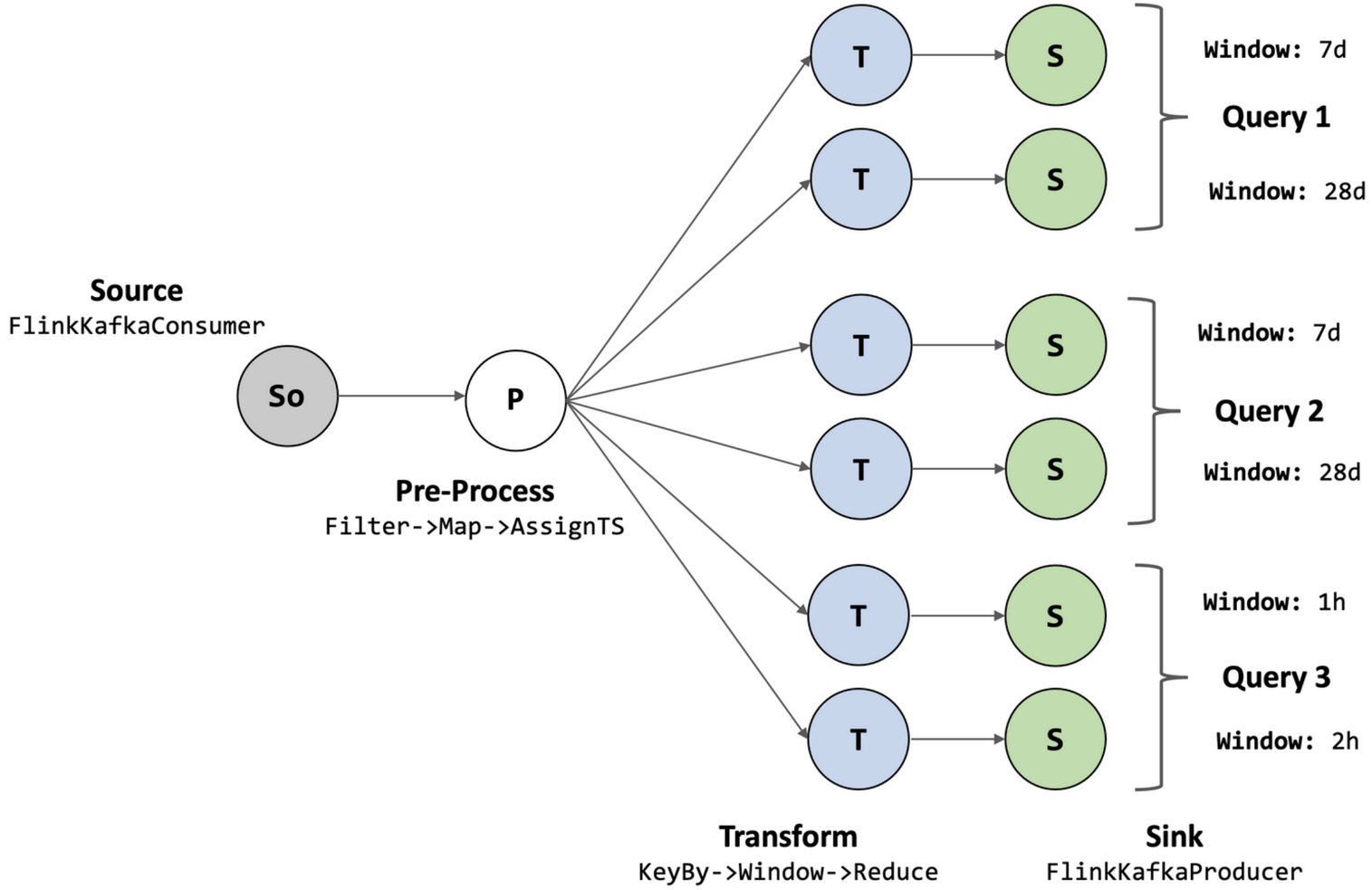
PUNTI CHIAVE

- Architettura Master-Worker
- Event time e Processing Time
- Stato Exaclty Once
- In-Memory
- Lateness
- Parallelismo

USO

- Processamento delle 3 query
- Produzione di metriche

Topologia


Source
FlinkKafkaConsumer
So
P
Pre-Process
Filter->Map->AssignTS
T
S
Window: 7d
Query 1
Window: 28d
Window: 7d
Query 2
Window: 28d
Window: 1h
Query 3
Window: 2h
Transform
KeyBy->Window->Reduce
Sink
FlinkKafkaProducer

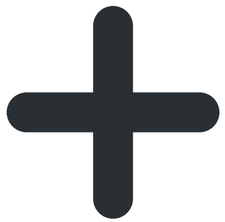
Applicativo Java

PRODUCER & CONSUMER

- **Ordina** il dataset
- **Replay** accelerato
- **Produce** messaggi per Kafka
- **Legge** messaggi da Kafka e li esporta in CSV

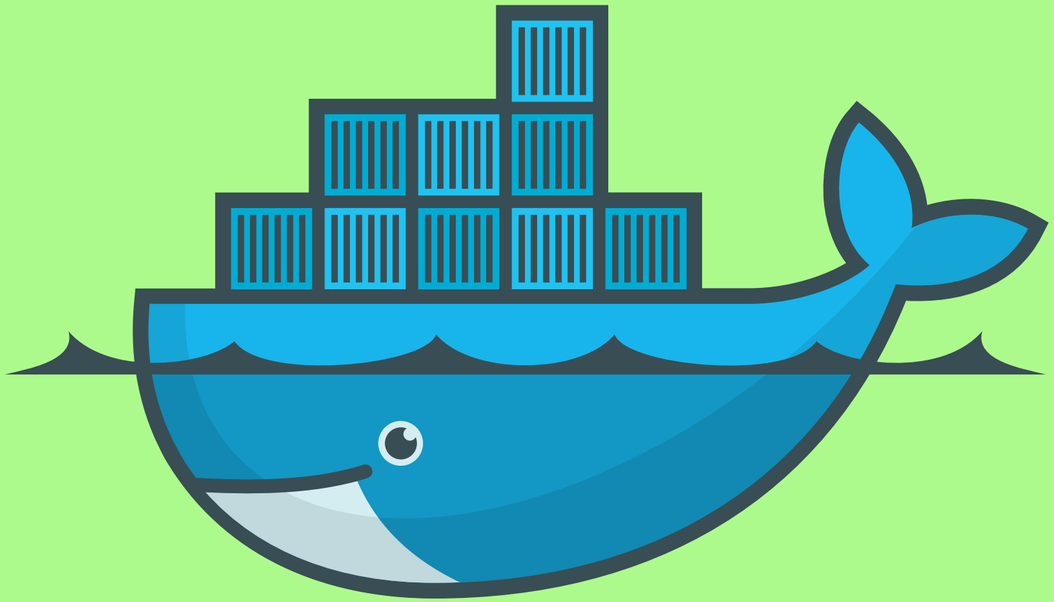
TOPOLOGY SUBMITTER

- **Estrae** grafo di esecuzione
- Comunica al **JobManager** il grafo e il **.jar** per le dipendenze grazie a **PackagedProgram**



UTILIZZATO PER

- **Dashboard** relativo a metriche
- **Monitorning** in tempo reale del cluster
- **Alarm** in situazioni di failure
- Si appogiano a **ROCKSDB**

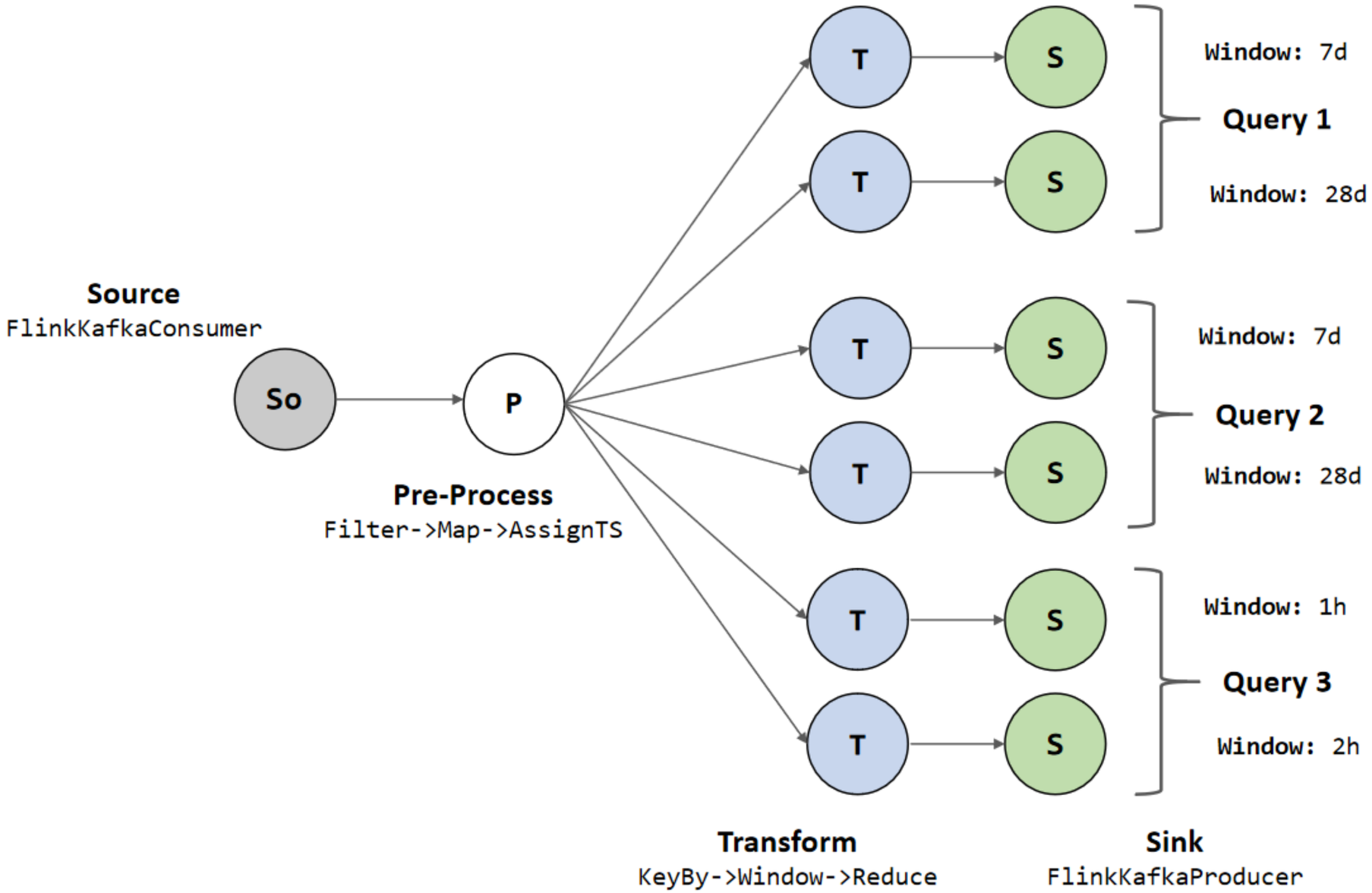


Deployment

DOCKER COMPOSE

- Simulazione di nodi della rete distinti
- Set-up e Clean-up automatico dei container
- Latenza pari a 0
- Risorse computazionali non ottimali

Queries overview



Preprocessing

AddSource
(per associare la source di Flink al topic di Kafka)

Filter
(per filtrare le tuple non appartenenti al mar Mediterraneo)

Map
(per associare ad ogni tupla la cella del mare corrispondente)

Map
(per associare ad ogni tupla la label 'occidentale' o orientale)

Map
(per associare ad ogni tupla il timeslot mattutino o pomeridiano)

Assign Timestamp AndWatermarks
(per definire quale campo delle tuple rappresentasse l'event time, definire una lateness di 5 minuti ed iniettare tuple di controllo [watermarks] nel flusso dei dati)

Details

- Divisione longitudinale (12.275) coincidente con il confine della cella ad ovest di Trapani (esclusione dell'Adriatico dal Mediterraneo occidentale)

- Lateness concessa 5 minuti così da coincidere con il tempo di emissione massimo di ogni segnale navale

Query 1

Filter
(per considerare solo il mediterraneo occidentale)

KeyBy
(per partizionare il flusso dei dati in base alla cella marittima)

Windows
(implementata come Tumbling)

Aggregate
(aggregatore e process window function per stato della finestra)

Map
(per il calcolo della media giornaliera e la costruzione della stringa di output)

Aggregator

- HashMap : (key , value) = (day + shipId , shipType)
- CreateAccumulator ---> ritorna nuova HashMap
- Add = HashMap.put(day + shipId , shipType)
- GetResult ---> array contente il conto degli shipType

ProcessWindowFunction

Permette di accedere informazioni sullo stato della finestra (la key ---> cella marittima ed il tempo di inizio della finestra) che poi vengono aggiunti all'output prodotto dalla funzione getResult dell'aggragatore

Query 1 risultato

Timestamp	Cell	SHIP_TYPE=35	Value	SHIP_TYPE=60-69	Value	SHIP_TYPE=70-79	Value	SHIP_TYPE=others	Value
2015/03/10	D1	SHIP_TYPE=35	0.0	SHIP_TYPE=60-69	0.0	SHIP_TYPE=70-79	0.0	SHIP_TYPE=others	0.42857143
2015/03/10	C1	SHIP_TYPE=35	0.0	SHIP_TYPE=60-69	0.0	SHIP_TYPE=70-79	0.0	SHIP_TYPE=others	0.14285715
2015/03/17	D1	SHIP_TYPE=35	0.0	SHIP_TYPE=60-69	0.0	SHIP_TYPE=70-79	0.0	SHIP_TYPE=others	1.0
2015/03/24	D1	SHIP_TYPE=35	0.0	SHIP_TYPE=60-69	0.0	SHIP_TYPE=70-79	0.0	SHIP_TYPE=others	1.0
2015/03/31	D1	SHIP_TYPE=35	0.0	SHIP_TYPE=60-69	0.0	SHIP_TYPE=70-79	0.0	SHIP_TYPE=others	1.1428572
2015/03/31	D2	SHIP_TYPE=35	0.0	SHIP_TYPE=60-69	0.0	SHIP_TYPE=70-79	0.0	SHIP_TYPE=others	0.2857143
2015/04/07	D1	SHIP_TYPE=35	0.0	SHIP_TYPE=60-69	0.0	SHIP_TYPE=70-79	0.0	SHIP_TYPE=others	1.4285715
2015/04/07	D2	SHIP_TYPE=35	0.0	SHIP_TYPE=60-69	0.0	SHIP_TYPE=70-79	0.0	SHIP_TYPE=others	0.71428573
2015/04/07	D3	SHIP_TYPE=35	0.0	SHIP_TYPE=60-69	0.0	SHIP_TYPE=70-79	0.0	SHIP_TYPE=others	0.42857143
2015/04/07	D5	SHIP_TYPE=35	0.0	SHIP_TYPE=60-69	0.0	SHIP_TYPE=70-79	0.0	SHIP_TYPE=others	0.2857143
2015/04/07	H8	SHIP_TYPE=35	0.0	SHIP_TYPE=60-69	0.0	SHIP_TYPE=70-79	0.0	SHIP_TYPE=others	0.14285715

Query 2

KeyBy
(basandosi sulla zona del mare 'occidentale' o 'orientele')

Windows
(implementata come Tumbling)

Aggregate
(aggregatore e process window function per stato della finestra)

Map
(per la formattazione dell'output)

Accumulator

- Custom class avente come attributi 2 HashMap, una per fascia oraria aventi struttura :
  (key , value) = (cella , Array<shipId + day >)

Aggregator

- CreateAccumulator ---> ritorna nuovo accumulator
- Add ---> in base alla fascia oraria accede all'HashMap corrispondente, quindi all'array associato a quella cella : se questo non contiene il valore (shipId + day), viene aggiunto all'array
- GetResult ---> ritorna un oggetto contenente 2 array ,uno per fascia oraria, contenenti la coppie : (cella , #navi)

ProcessWindowFunction

- Permette di accedere informazioni sullo stato della finestra (la key ---> area marittima ed il tempo di inizio della finestra).

- Computa per ogni coppia di array generate dal getResult, le 3 celle più frequentate di ogni array. Quindi produce il nuovo output unendo a queste informazioni quelle estratte dalla finestra.

Query 2 risultato

Timestamp	Sea	Time slot	Leaderboard	Time slot	Leaderboard
2015/04/28	WESTERN_MEDITERANEAN_SEA	BEFORE_NOON	F9;J14;F8;	AFTER_NOON	J14;H8;F9;
2015/04/28	EASTERN_MEDITERANEAN_SEA	BEFORE_NOON	C20;D20;G33;	AFTER_NOON	C20;D20;G33;
2015/05/05	EASTERN_MEDITERANEAN_SEA	BEFORE_NOON	C20;D20;G33;	AFTER_NOON	C20;D20;G33;
2015/05/05	WESTERN_MEDITERANEAN_SEA	BEFORE_NOON	F9;F8;H8;	AFTER_NOON	H8;F8;G8;
2015/05/12	WESTERN_MEDITERANEAN_SEA	BEFORE_NOON	F9;F8;H8;	AFTER_NOON	F8;H8;F7;
2015/05/12	EASTERN_MEDITERANEAN_SEA	BEFORE_NOON	G33;C20;D20;	AFTER_NOON	G33;C20;D20;

Query 3

KeyBy
(basandosi sul campo tripId)

Windows
(implementata come Tumbling)

Aggregate
(aggregatore e process window function per stato della finestra)

KeyBy
(basandosi timeStamp)

Windows
(implementata come Tumbling)

Aggregate
(aggregatore)

Map
(per la formattazione dell'output)

Aggregator

- Tuple5 contenente < TripId, OldTs, CurrentTs, Lon, Lat>
- CreateAccumulator ---> ritorna nuova Tuple5
- Add ---> confronto Ts con OldTs :
  - Se precedente ripopola solo il campo OldTs dell'accumulator ed inserisce i suoi valori di Lat e Lon in un HashMap globale <TripId, Tuple3<Ts, Lon, Lat>>
  - Se successivo si confronta con il CurrentTs dell'accumulator e se successivo anche a questo ripopola tutti i campi dell'accumulator eccetto OldTs con i suoi valori
- GetResult ---> ritorna la distanza euclidea tra Lon e Lat riportati nell'accumulator (che corrispondono all'ultima posizione vista per quel viaggio) ed i valori di Lon e Lat iniziali mantenuti nell'HashTable globale

ProcessWindowFunction

Permette di accedere informazioni sullo stato della finestra (la key ---> il tripId ed il tempo di inizio della finestra) che poi vengono aggiunti all'output prodotto dalla funzione getResult dell'aggragatore

Aggregator

- Accumulatore custom contenente un campo Ts per il valore della finestra e due array, uno per gli score ed uno per i tripId corrispondenti
- CreateAccumulator ---> ritorna nuova accumulatore custom
- Add ---> per ogni tupla emessa dalla finestra precedente ne confronta lo score di navigazione con il minimo dell'array mantenuto nell'accumulatore e se risulta maggiore lo sostituisce a quest'ultimo
- GetResult ---> ritorna gli array ordinati contenenti i 5 score più grandi e relativi tripId e li concatena insieme al campo Ts a formare l'output

Details

- Offset : parametro delle finestre che ne ha permesso l'allineamento con il timestamp della prima tupla vista

- Query 3 :
  - HashMap globale
  - Parallelismo 1 dell'ultimo operatore

Query 3 risultato

Timestamp	TripID 1	Score	TripID 2	Score	TripID 3	Score	TripID 4	Score	TripID 5	Score
2015/04/11 07:15	0xccfc6_09-04-15 13:	8.679045	0x4a019_09-04-15 6:>	7.362404	0xbc56a_10-04-15 12	3.2201712	0x6d679_11-04-1	7.8436814E-4	0xc49c8_10-04-1	9.620773E-5
2015/04/11 09:15	0xccfc6_09-04-15 13:	8.720733	0xbc56a_10-04-15 12	3.5111074	0x6d679_11-04-15 2:	0.0027859646	0xccfc6_11-04-15	1.19158016E-	0xc49c8_10-04-1	9.620773E-5
2015/04/11 11:15	0x4a019_09-04-15 6:>	8.294712	0xbc56a_10-04-15 12	3.8169024	0x6d679_11-04-15 2:	0.0031363505	0xc49c8_10-04-1	8.9564186E-5	0xccfc6_11-04-15	5.521488E-5
2015/04/11 13:15	0x4a019_09-04-15 6:>	8.763398	0xbc56a_10-04-15 12	4.099406	0xec4e1_11-04-15 14	0.30265495	0x6d679_11-04-1	0.0036652423	0xc49c8_10-04-1	8.538981E-5
2015/04/11 15:15	0x4a019_09-04-15 6:>	8.880867	0xbc56a_10-04-15 12	4.3926606	0xec4e1_11-04-15 14	0.8389703	0x6d679_11-04-1	0.0028012453	0xc49c8_10-04-1	9.421607E-5
2015/04/11 17:15	0xbc56a_10-04-15 12	4.732138	0xec4e1_11-04-15 14	1.3847756	0xc49c8_11-04-15 18	0.19521758	0x6d679_11-04-1	0.003011305	0xc49c8_10-04-1	9.993123E-5
2015/04/11 19:15	0xbc56a_10-04-15 12	5.058954	0xec4e1_11-04-15 14	1.8488048	0xc49c8_11-04-15 18	0.31091502	0x6d679_11-04-1	0.0023217907	0xccfc6_11-04-15	6.1161685E-5
2015/04/11 21:15	0xbc56a_10-04-15 12	5.3790507	0xec4e1_11-04-15 14	2.402895	0xe4728_09-04-15 0:	2.0017064	0xc49c8_11-04-1	0.28971836	0x6d679_11-04-1	0.0038236296
2015/04/11 23:15	0xbc56a_10-04-15 12	5.7072253	0xec4e1_11-04-15 14	2.8007953	0xc49c8_11-04-15 18	0.49367702	0x0e26c_11-04-1	0.10576931	0x6d679_11-04-1	0.0038311537
2015/04/12 01:15	0xbc56a_10-04-15 12	6.049285	0xc49c8_11-04-15 18	0.50137395	0x0e26c_11-04-15 5:	0.38032144	0x6d679_11-04-1	0.0038792088	0xccfc6_11-04-15	7.0470334E-5
2015/04/12 03:15	0xbc56a_10-04-15 12	6.435879	0x0e26c_11-04-15 5:>	0.47318697	0x6d679_11-04-15 2:	0.0032768033	0xccfc6_11-04-15	5.867229E-5		
2015/04/12 05:15	0x4a019_09-04-15 6:>	11.611907	0xbc56a_10-04-15 12	6.7986064	0x6d679_11-04-15 2:	0.003948462	0xccfc6_11-04-15	5.059195E-5		
2015/04/12 07:15	0x4a019_09-04-15 6:>	11.974395	0xbc56a_10-04-15 12	7.1048846	0xec4e1_11-04-15 14	4.7966957	0x6d679_11-04-1	0.0039810184	0xccfc6_11-04-15	9.191245E-5
2015/04/12 09:15	0x4a019_09-04-15 6:>	12.332581	0xbc56a_10-04-15 12	7.36404	0xec4e1_11-04-15 14	5.110301	0xc49c8_11-04-1	1.9438314	0x6d679_11-04-1	0.0038875628

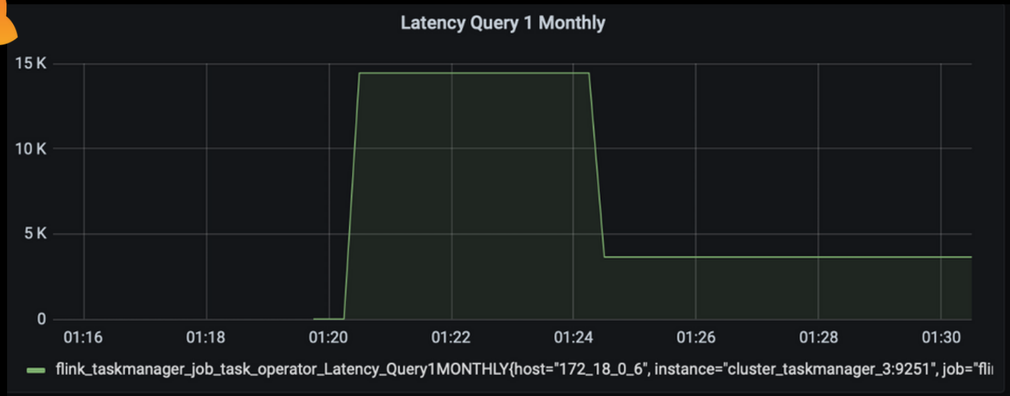
METRICHE

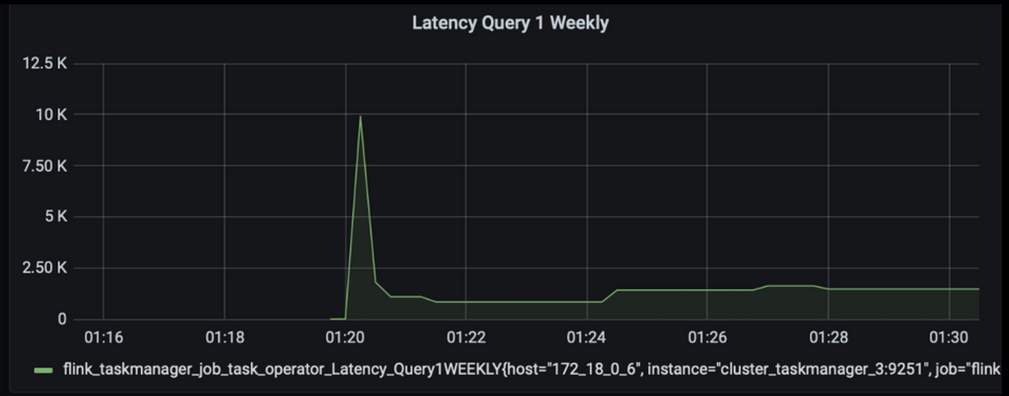
Perchè

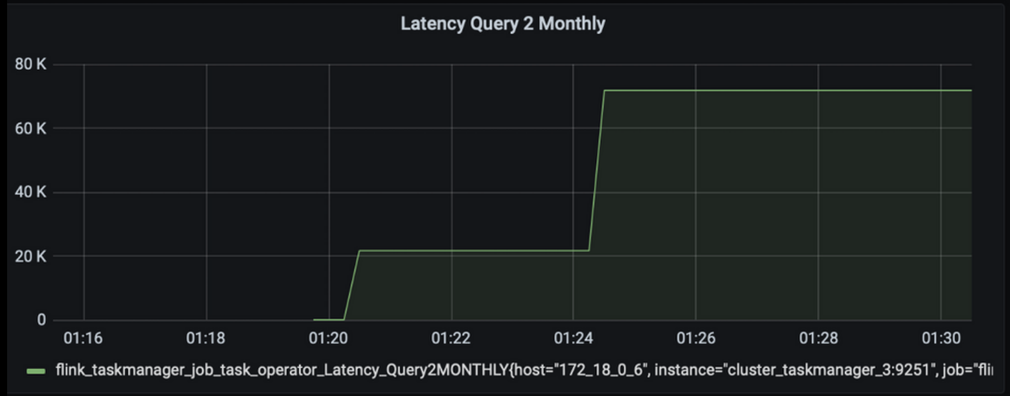
Misurare l'andamento di prestazioni di Flink e *healthy monitoring*

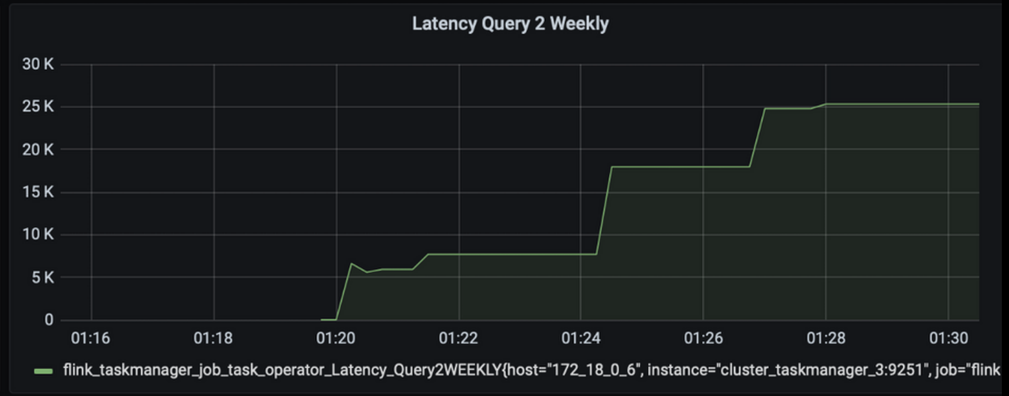
Come

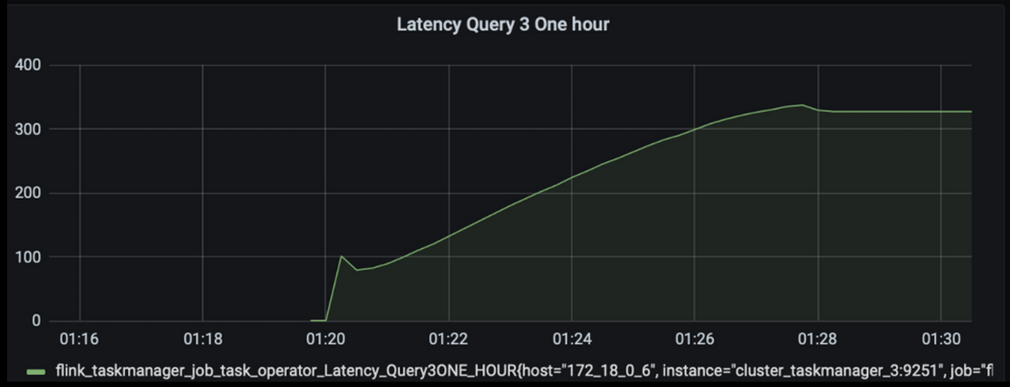
Metriche Built-In di Flink
Prometheus w/ Oshi-Core e JNA
RocksDB

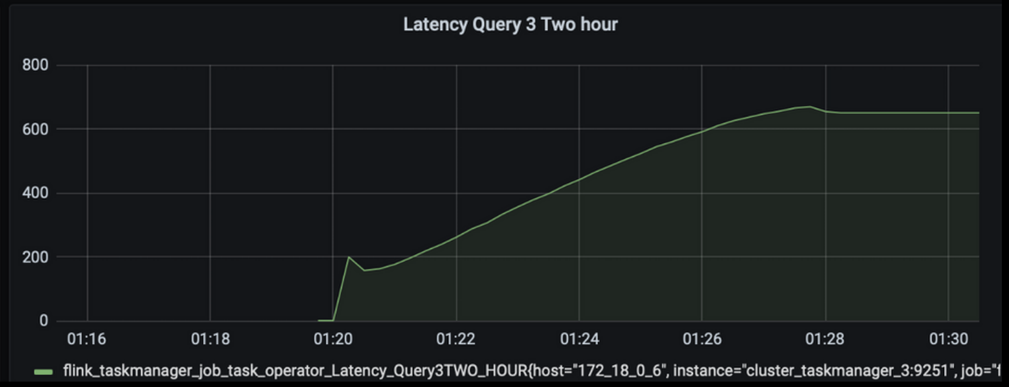

Latency Query 1 Monthly
15 K
10 K
5 K
0
01:16
01:18
01:20
01:22
01:24
01:26
01:28
01:30
flink_taskmanager_job_task_operator_Latency_Query1MONTHLY{host="172_18_0_6", instance="cluster_taskmanager_3:9251", job="fli

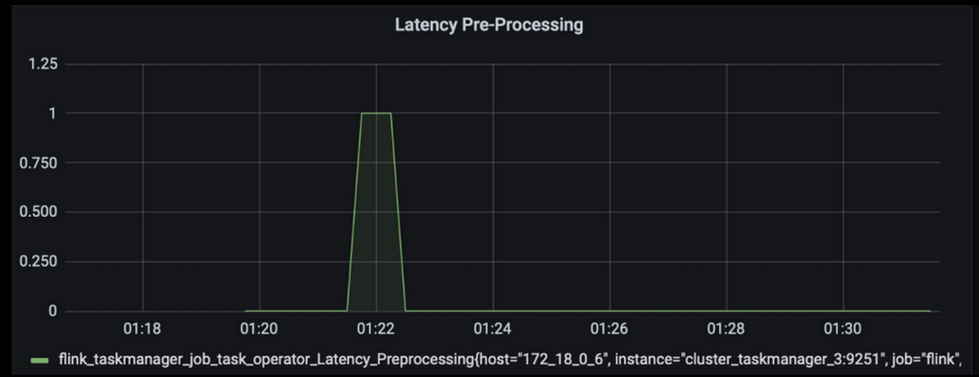

Latency Query 1 Weekly
12.5 K
10 K
7.50 K
5 K
2.50 K
0
01:16
01:18
01:20
01:22
01:24
01:26
01:28
01:30
flink_taskmanager_job_task_operator_Latency_Query1WEEKLY{host="172_18_0_6", instance="cluster_taskmanager_3:9251", job="flink

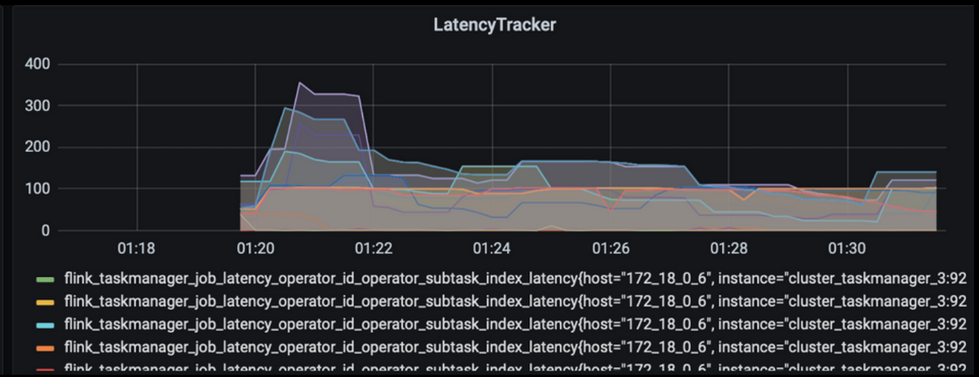

Latency Query 2 Monthly
80 K
60 K
40 K
20 K
0
01:16
01:18
01:20
01:22
01:24
01:26
01:28
01:30
flink_taskmanager_job_task_operator_Latency_Query2MONTHLY{host="172_18_0_6", instance="cluster_taskmanager_3:9251", job="fli

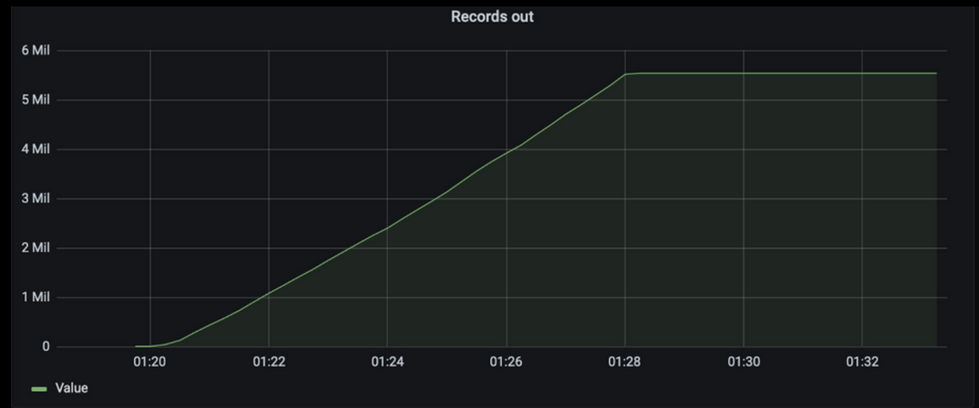

Latency Query 2 Weekly
30 K
25 K
20 K
15 K
10 K
5 K
0
01:16
01:18
01:20
01:22
01:24
01:26
01:28
01:30
flink_taskmanager_job_task_operator_Latency_Query2WEEKLY{host="172_18_0_6", instance="cluster_taskmanager_3:9251", job="flink

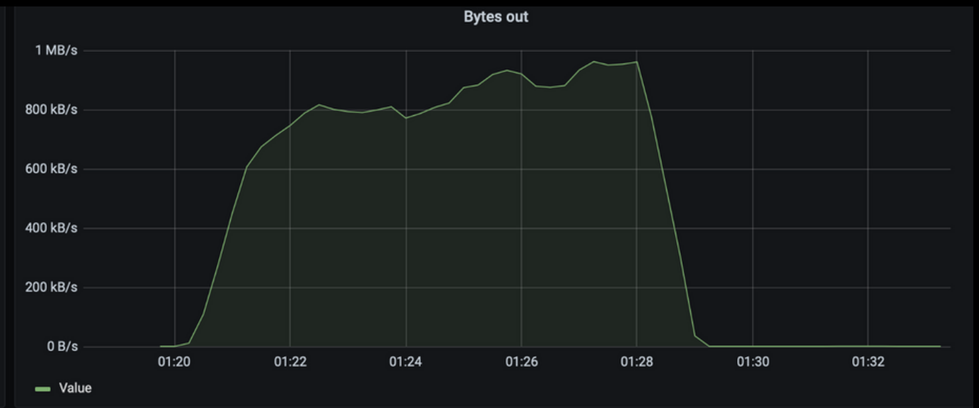

Latency Query 3 One hour
400
300
200
100
0
01:16
01:18
01:20
01:22
01:24
01:26
01:28
01:30
flink_taskmanager_job_task_operator_Latency_Query3ONE_HOUR{host="172_18_0_6", instance="cluster_taskmanager_3:9251", job="fl

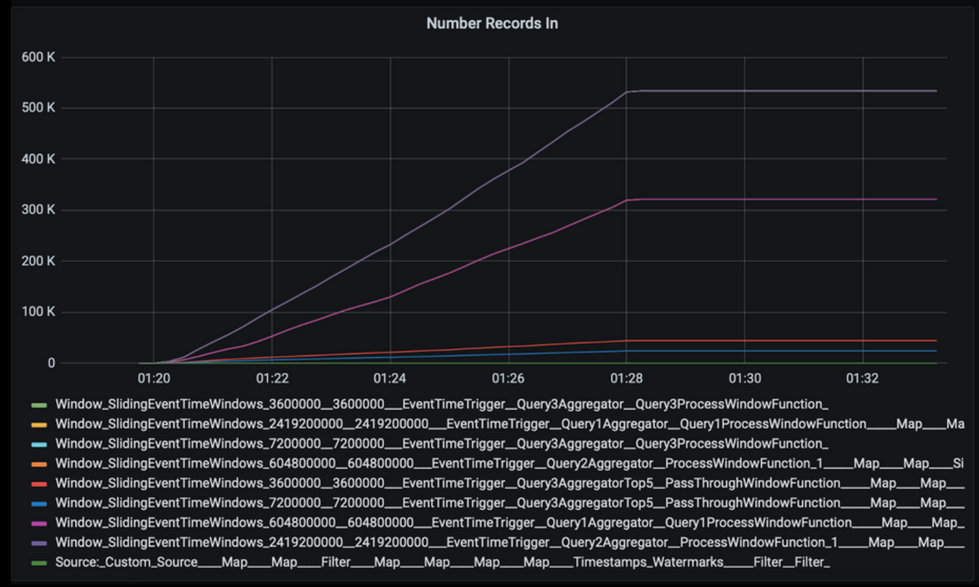

Latency Query 3 Two hour
800
600
400
200
0
01:16
01:18
01:20
01:22
01:24
01:26
01:28
01:30
flink_taskmanager_job_task_operator_Latency_Query3TWO_HOUR{host="172_18_0_6", instance="cluster_taskmanager_3:9251", job="f

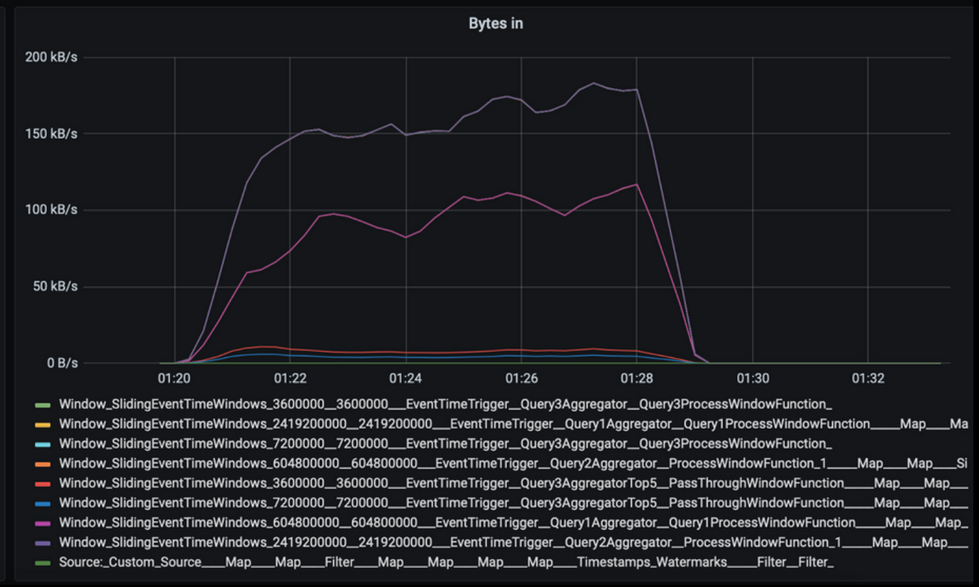

Latency Pre-Processing
1.25
1
0.750
0.500
0.250
0
01:18
01:20
01:22
01:24
01:26
01:28
01:30
flink_taskmanager_job_task_operator_Latency_Preprocessing{host="172_18_0_6", instance="cluster_taskmanager_3:9251", job="flink",


LatencyTracker
400
300
200
100
0
01:18
01:20
01:22
01:24
01:26
01:28
01:30
flink_taskmanager_job_latency_operator_id_operator_subtask_index_latency{host="172_18_0_6", instance="cluster_taskmanager_3:92
flink_taskmanager_job_latency_operator_id_operator_subtask_index_latency{host="172_18_0_6", instance="cluster_taskmanager_3:92
flink_taskmanager_job_latency_operator_id_operator_subtask_index_latency{host="172_18_0_6", instance="cluster_taskmanager_3:92
flink_taskmanager_job_latency_operator_id_operator_subtask_index_latency{host="172_18_0_6", instance="cluster_taskmanager_3:92


Records out
6 Mil
5 Mil
4 Mil
3 Mil
2 Mil
1 Mil
0
01:20
01:22
01:24
01:26
01:28
01:30
01:32
Value

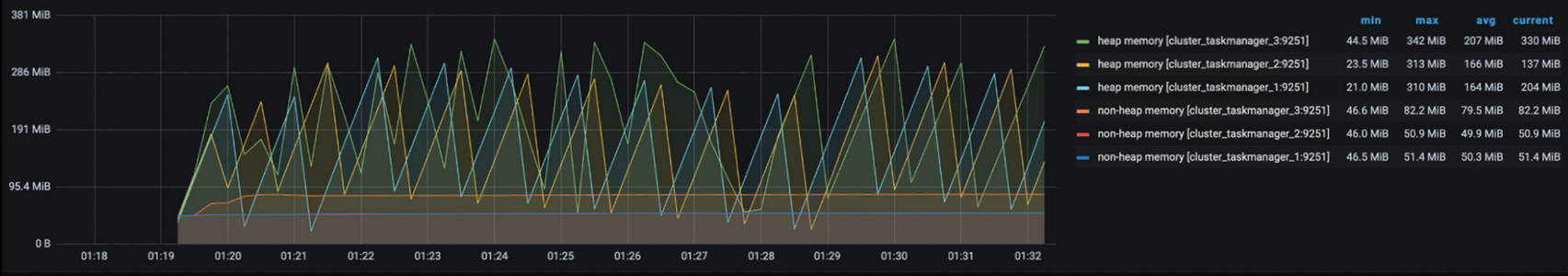

Bytes out
1 MB/s
800 kB/s
600 kB/s
400 kB/s
200 kB/s
0 B/s
01:20
01:22
01:24
01:26
01:28
01:30
01:32
Value


Number Records In
600 K
500 K
400 K
300 K
200 K
100 K
0
01:20
01:22
01:24
01:26
01:28
01:30
01:32
Window_SlidingEventTimeWindows_3600000__3600000___EventTimeTrigger__Query3Aggregator__Query3ProcessWindowFunction_
Window_SlidingEventTimeWindows_2419200000__2419200000___EventTimeTrigger__Query1Aggregator__Query1ProcessWindowFunction____Map____Ma
Window_SlidingEventTimeWindows_7200000__7200000___EventTimeTrigger__Query3Aggregator__Query3ProcessWindowFunction_
Window_SlidingEventTimeWindows_604800000__604800000___EventTimeTrigger__Query2Aggregator__ProcessWindowFunction_1____Map____Map____Si
Window_SlidingEventTimeWindows_3600000__3600000___EventTimeTrigger__Query3AggregatorTop5__PassThroughWindowFunction____Map____Map___
Window_SlidingEventTimeWindows_7200000__7200000___EventTimeTrigger__Query3AggregatorTop5__PassThroughWindowFunction____Map____Map___
Window_SlidingEventTimeWindows_604800000__604800000___EventTimeTrigger__Query1Aggregator__Query1ProcessWindowFunction____Map____Map_
Window_SlidingEventTimeWindows_2419200000__2419200000___EventTimeTrigger__Query2Aggregator__ProcessWindowFunction_1____Map____Map___
Source:_Custom_Source____Map____Map____Filter____Map____Map____Map____Map____Timestamps_Watermarks____Filter__Filter_

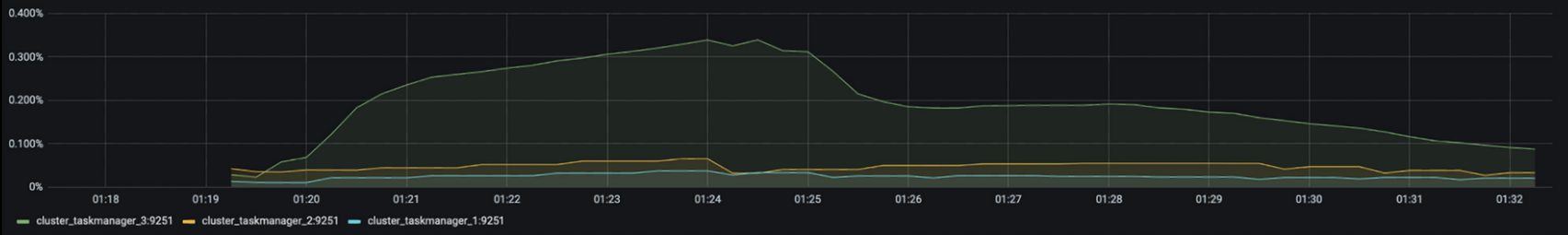

Bytes in
200 kB/s
150 kB/s
100 kB/s
50 kB/s
0 B/s
01:20
01:22
01:24
01:26
01:28
01:30
01:32
Window_SlidingEventTimeWindows_3600000__3600000___EventTimeTrigger__Query3Aggregator__Query3ProcessWindowFunction_
Window_SlidingEventTimeWindows_2419200000__2419200000___EventTimeTrigger__Query1Aggregator__Query1ProcessWindowFunction____Map____Ma
Window_SlidingEventTimeWindows_7200000__7200000___EventTimeTrigger__Query3Aggregator__Query3ProcessWindowFunction_
Window_SlidingEventTimeWindows_604800000__604800000___EventTimeTrigger__Query2Aggregator__ProcessWindowFunction_1____Map____Map____Si
Window_SlidingEventTimeWindows_3600000__3600000___EventTimeTrigger__Query3AggregatorTop5__PassThroughWindowFunction____Map____Map___
Window_SlidingEventTimeWindows_7200000__7200000___EventTimeTrigger__Query3AggregatorTop5__PassThroughWindowFunction____Map____Map___
Window_SlidingEventTimeWindows_604800000__604800000___EventTimeTrigger__Query1Aggregator__Query1ProcessWindowFunction____Map____Map_
Window_SlidingEventTimeWindows_2419200000__2419200000___EventTimeTrigger__Query2Aggregator__ProcessWindowFunction_1____Map____Map___
Source:_Custom_Source____Map____Map____Filter____Map____Map____Map____Map____Timestamps_Watermarks____Filter__Filter_

TaskManager

Memory used

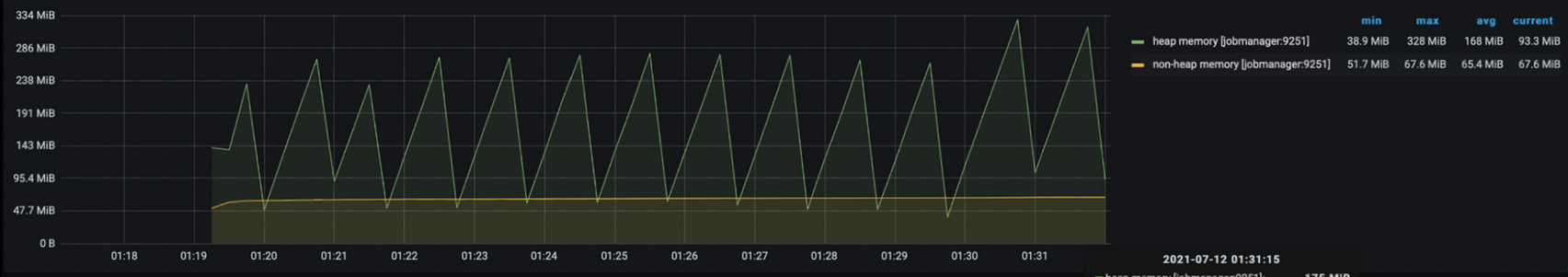


GC Overhead

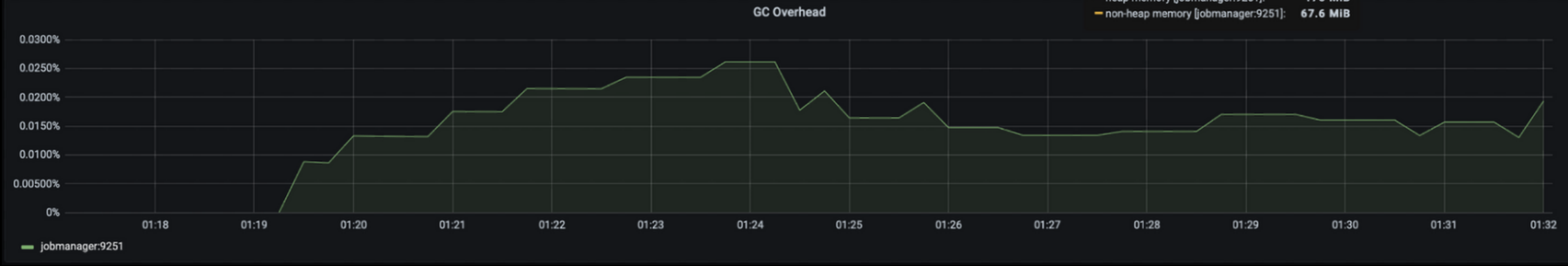


JobManager

Memory used



GC Overhead



SVILUPPI FUTURI

MAPE

per gestire carico variabile

Process Window

Rimozione ridurre consumo memoria

Occidentale e Orientale

Suddivisione a grana fine del Mare

Memoria

Reset HashMap 3° Query

CREDITS

Andrea Paci

andrea.paci1998@gmail.com

Alessandro Amici

a.amici@outlook.it

Repository GitHub

https://github.com/andreapaci/SABD2