Savitar: Analisi di dati in tempo reale di una partita di calcio

Sistemi e architetture per Big Data

Outline

- Scopo del progetto e descrizione dataset.
- Scelta del framework.
- Descrizione delle query.
- Valutazione delle prestazioni.

Scopo e obiettivi del progetto

Lo scopo del progetto è di analizzare in tempo reale, tramite un framework open-source di data stream processing il dataset del **DEBS 2013 Grand Challenge** riguardante una ***partita di calcio***, rispondendo ad alcune query rilevanti per gli allenatori delle due squadre e per gli spettatori della partita.

Scelta del framework

Tabella di confronto tra i framework.

	Storm	**Spark Streaming**	**Flink**
Modello Streaming	Nativo	Micro-Batching	Nativo
Fault-tolerance	At-least once	Exactly once	ExactOnce
Latenza	Bassa	Media	bassa
Throughput	Bassa	Alta	Alta
Maturità	Alta	Alta	Media

Scelta del framework

Abbiamo deciso di implementare il nostro progetto usando Apache Storm.

I fattori determinanti sono stati:

- La bassa latenza
- La maturità del progetto
- Il processamento a livello di singola tupla (no micro-batch)

Scelta del framework

I fattori di esclusione sono stati:

- Nonostante ***Heron*** sia il progetto destinato a sostituire Storm, garantendo latenza più bassa, throughput più alto, migliore gestione delle risorse… tuttavia è ancora un **progetto in Beta**
- ***Spark Streaming*** lavora a **micro-batch**, introducendo una latenza non indifferente
- Volendo **confrontare** due diversi framework con gli altri gruppi, abbiamo scartato ***Flink***.

Descrizione del dataset

Il dataset riguarda i dati acquisiti tramite sensori wireless durante una partita di calcio tra 2 squadre da 8 giocatori che emettono dati:

- Alla frequenza di 200Hz. (Ogni giocatore ha 2 sensori)
- Con il seguente schema
  - sid, ts, x, y, z, |v|, |a|, vx, vy, vz, ax, ay, az

Ci sono anche i dati emessi dai guanti del portiere e dal pallone che nelle nostre query non sono stati considerati, ma che fanno parte del dataset che consideriamo.

Eseguire il progetto

Il progetto può essere eseguito e configurato con semplicità.

Si può effettuare il run della topologia e specificare tramite argomenti di riga di comando quali parametri cambiare rispetto a quelli di default.

Per eseguire il programma si devono passare gli argomenti relativi alla posizione dei file contenenti i dati da processare.

Query uno

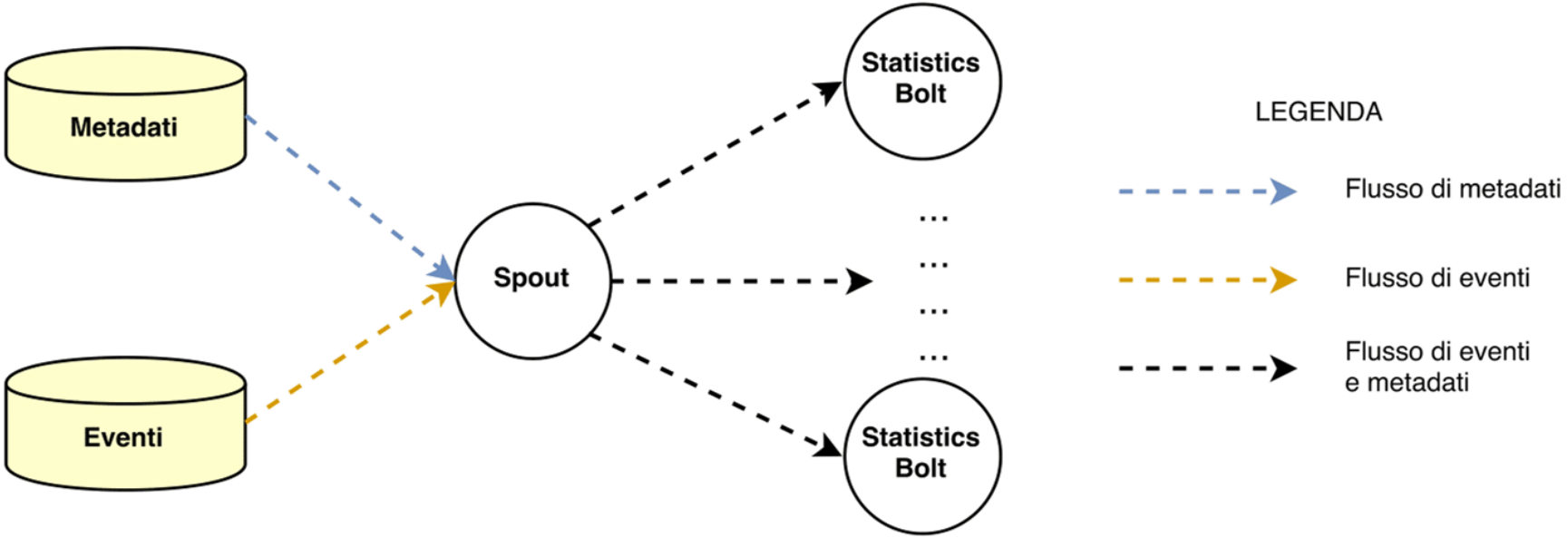
Analizzare le ***prestazioni nella corsa di ogni giocatore*** che partecipa alla partita:

- Totale distanza percorsa dal giocatore.
- Velocità media del giocatore.

Tali statistiche sono calcolate per diverse finestre temporali:

- 1 minuto
- 5 minuti
- intera partita

Query Uno: Topologia



Query uno: Spout

- Lo spout legge prima dal file metadata ed inoltra in field grouping a tutti i bolt le informazioni relative a quali sono i sensori di un giocatore.
- In seguito si occupa di leggere dal file che contiene tutti i dati dei sensori e inoltrarli verso il bolt statistics tramite fieldsGrouping fatto sull'id del giocatore.

Query uno: Statistics Bolt

- Lo statistics Bolt si occupa di gestire:
  - messaggi di configurazione, legati al file metadata
  - messaggi di tipo evento.
- Quando arriva un nuovo evento generato da un sensore:
  - Si controlla se il tempo è avanzato talmente tanto da dover emettere in output nuove statistiche
  - Si gestisce il dato generato dal sensore aggiornando la distanza percorsa dal giocatore.

Query uno: Frequenza di emissione delle statistiche

Le statistiche vengono calcolate per 3 intervalli di tempi, ma vengono emesse ad intervalli regolari con una frequenza che viene decisa in fase di configurazione della topologia.

Le statistiche potrebbero essere emesse ogni 15 secondi, il che significherebbe che ogni 15 secondi conosceremo le statistiche relative all'ultimo minuto, agli ultimi 5 minuti e all'intera partita già giocata.

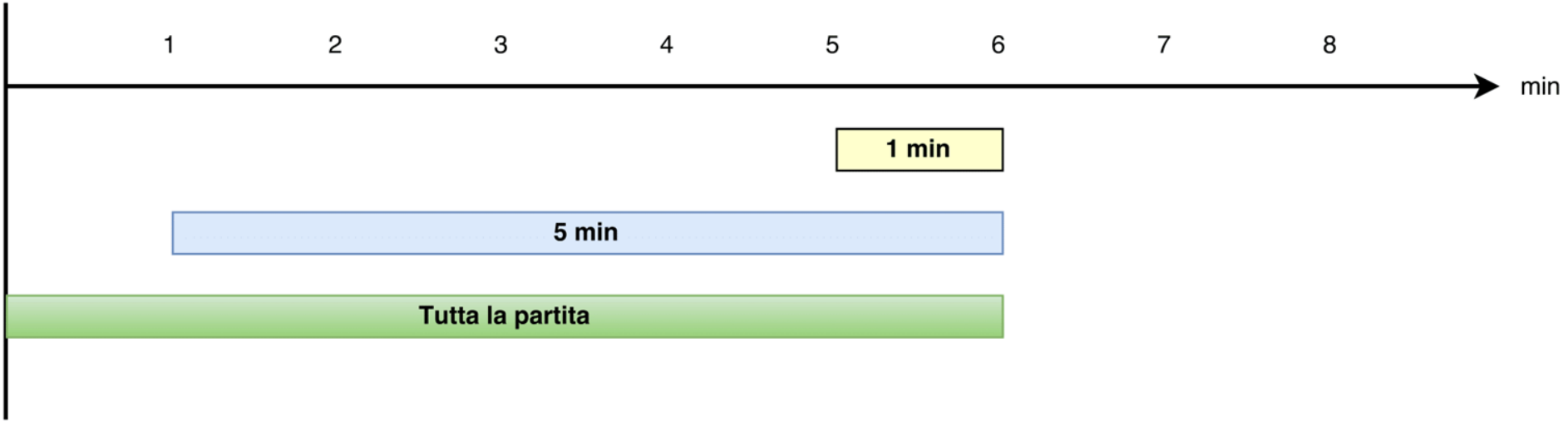
Query uno: Algoritmo legato all'intervallo di emissione

Quando arriva un evento ci troveremo tra una emissione e la successiva che avverrà in un istante pari a maxTime. Per cui l'algoritmo prevede di:

- se l'evento ha timestamp > maxTime
  - Emettere tutte le statistiche aggiornate
- Aggiornare i dati raccolti in base all'evento ricevuto:
  - Si calcola la distanza percorsa dal giocatore.
  - Si aggiorna la distanza del giocatore.

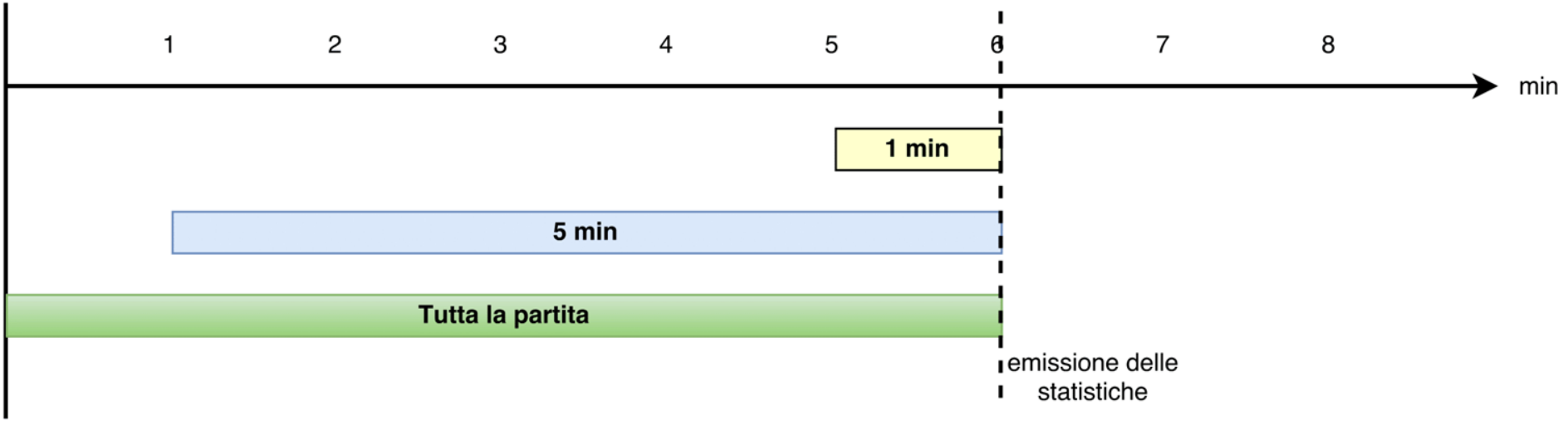
Esempio

Esempio per capire come lavorano le finestre temporali



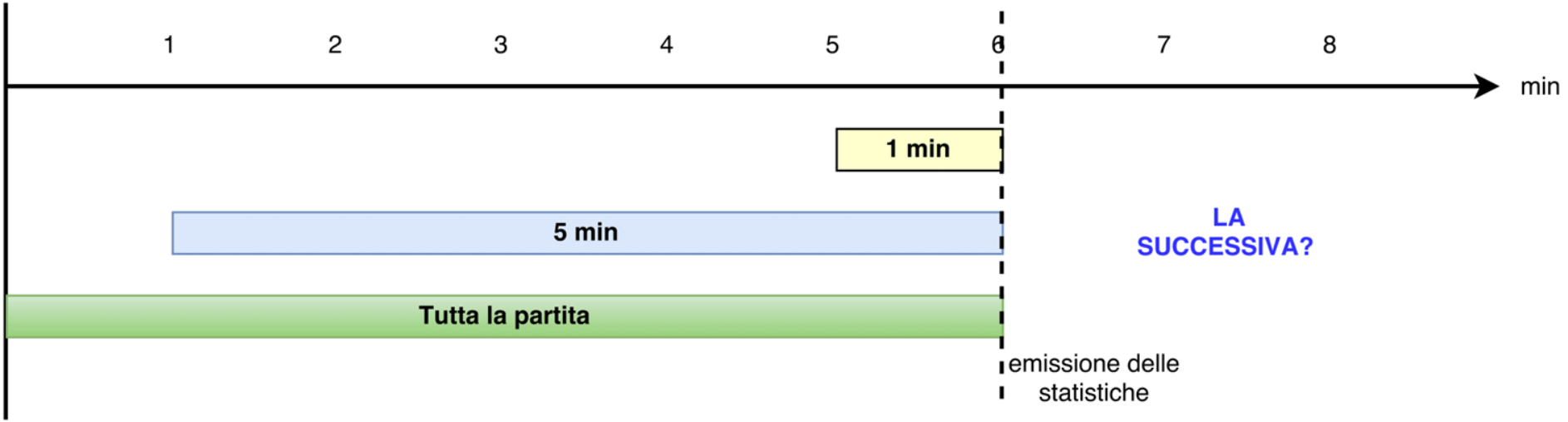
Esempio

Esempio per capire come lavorano le finestre temporali



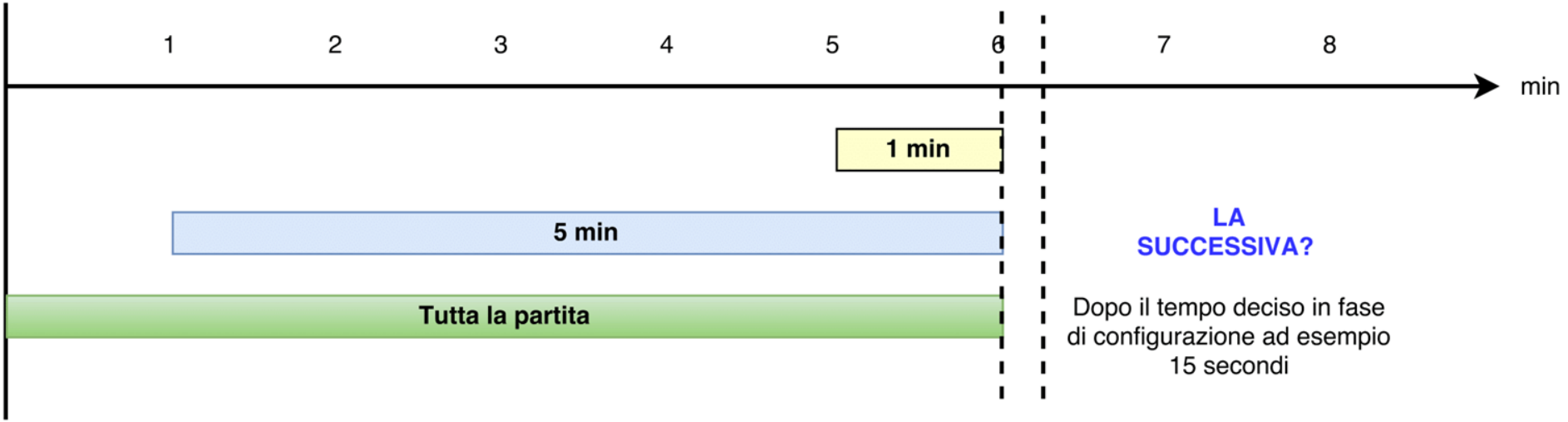
Esempio

Esempio per capire come lavorano le finestre temporali



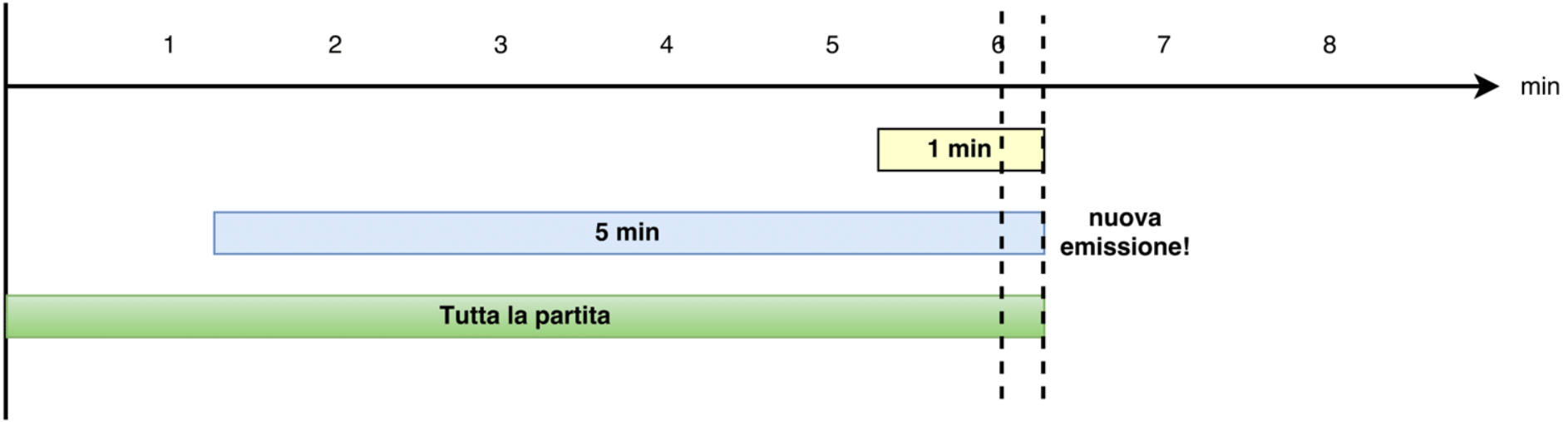
Esempio

Esempio per capire come lavorano le finestre temporali



Esempio

Esempio per capire come lavorano le finestre temporali



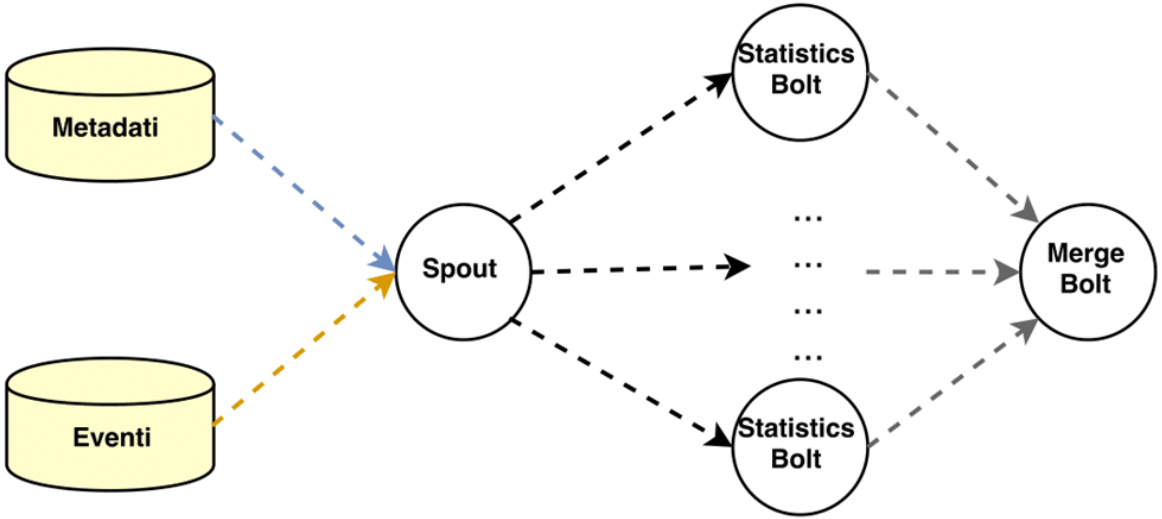
Query due

Si richiede di fornire la classifica aggiornata in tempo reale dei ***5 giocatori più veloci***.

Tali statistiche dovranno essere calcolate per diverse finestre temporali in modo tale da permettere di confrontare le prestazioni dei giocatori più veloci durante lo svolgimento della partita. Le finestre temporali hanno durata di:

- 1 minuto.
- 5 minuti.
- Intera partita.

Query due: Topologia



Query due

L’elemento innovativo della query due è il mergeBolt:

- Riceve le statistiche relative ad alcune finestre temporali specifiche.
- Ordina gli elementi e poi emette i primi 5 risultati.

Query Tre

L’obiettivo della terza query è di calcolare le statistiche relative a quanto tempo **ciascun giocatore** trascorre nelle diverse zone del campo di gioco.

A tale scopo, si suddivide il campo di gioco in una griglia di celle di uguale dimensione, con 8 celle lungo l’asse x e 13 celle lungo l’asse y per un totale di 104 celle.

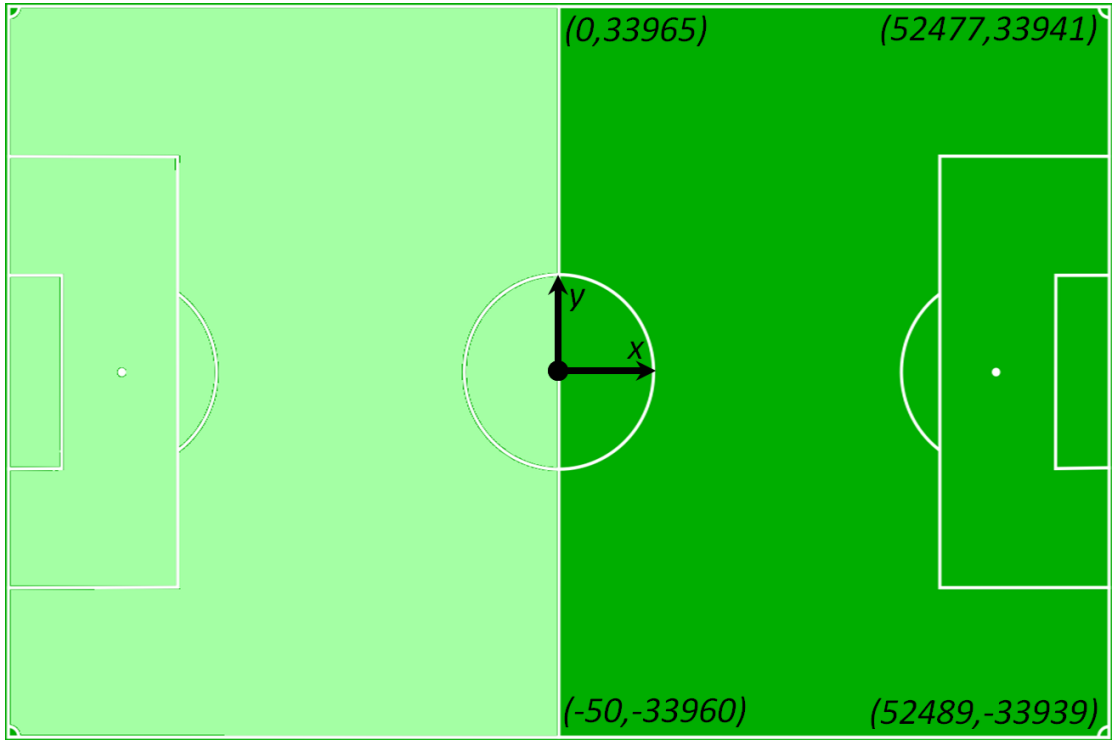
Query Tre

Tali statistiche dovranno essere calcolate per diverse finestre temporali in modo da fornire per ciascun giocatore la percentuale di tempo che il giocatore trascorre in ciascuna cella.

Le differenti finestre temporali sono:

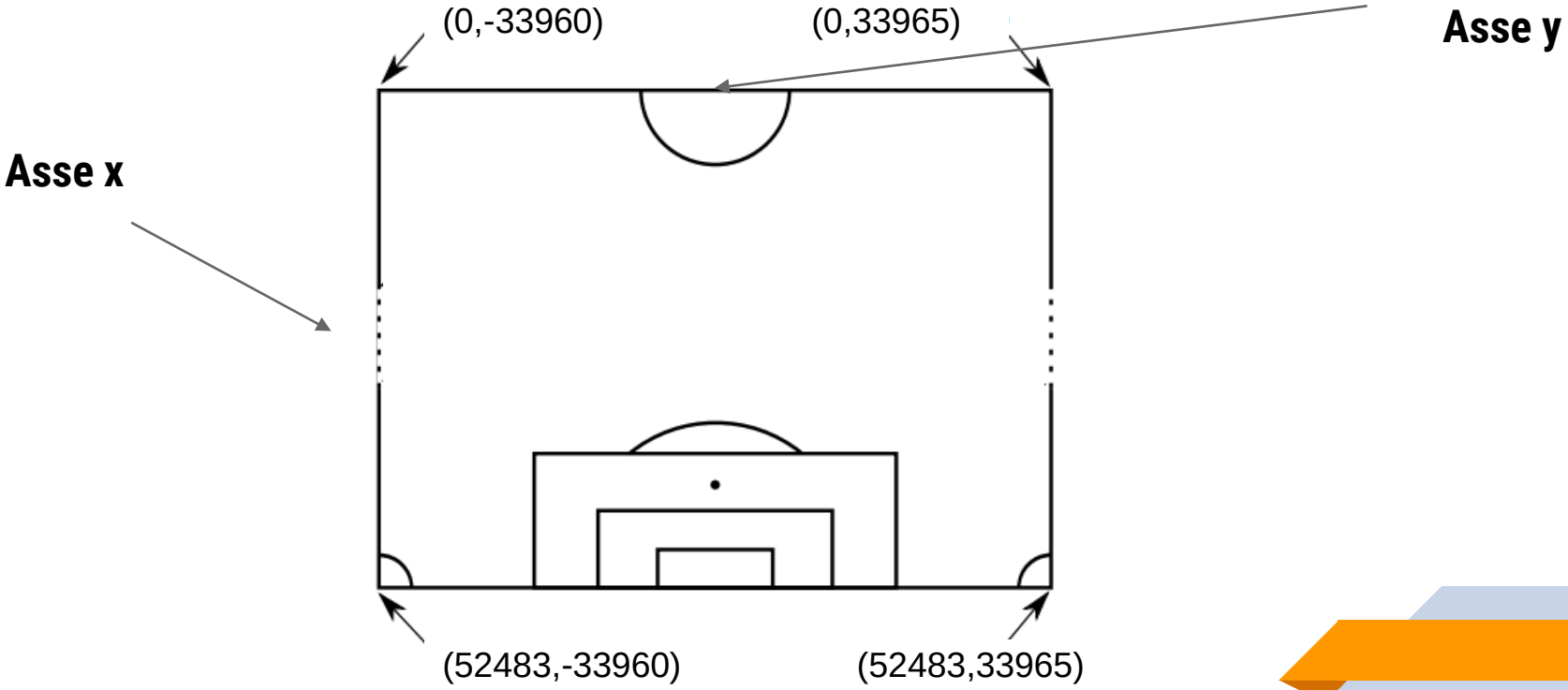
- 1 minuto.
- l’intera partita.

Query Tre: il campo

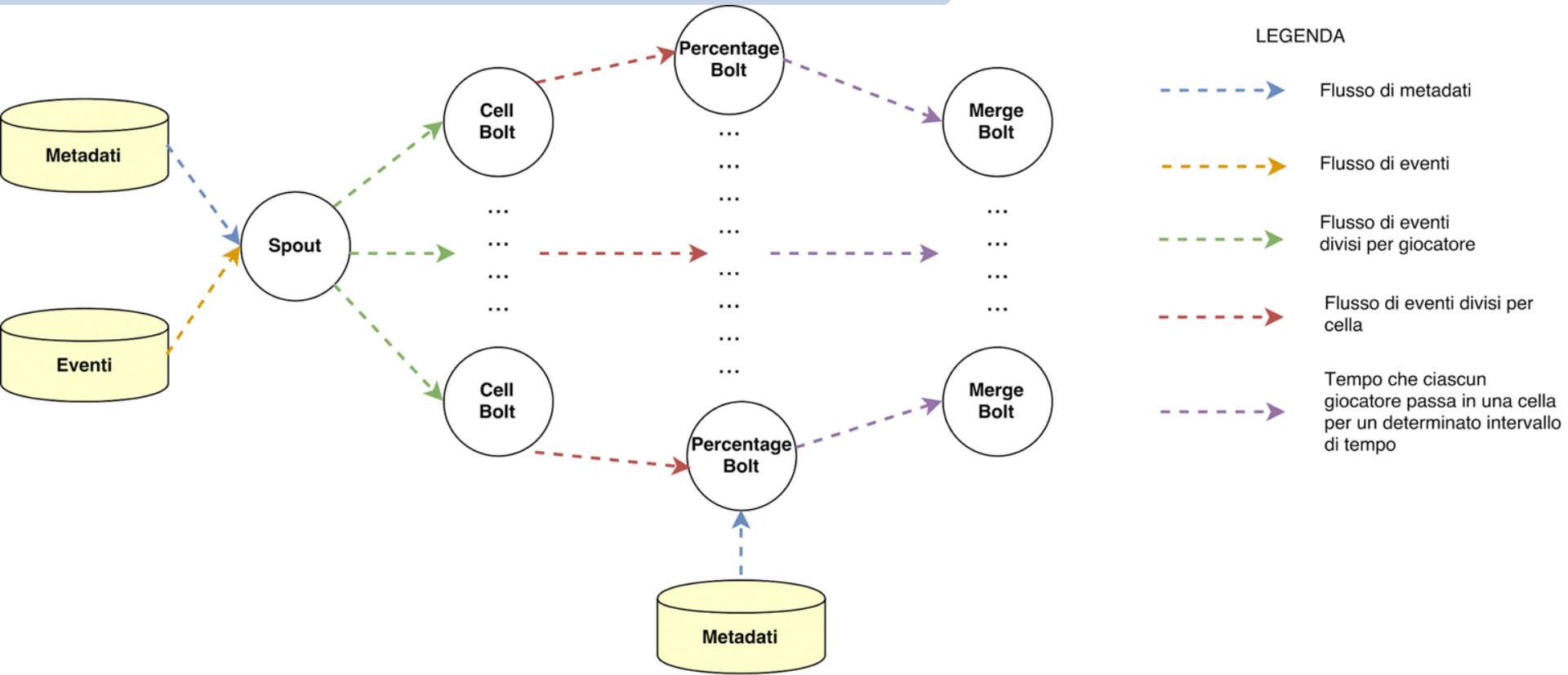


Come notiamo dalla figura il campo non è un rettangolo perfetto, possiamo quindi approssimare usando le seguenti coordinate:
(0, 33965),
(0, −33960),
(52483, 33965)
(52483, −33960).

Query Tre: il campo



Query Tre: Topologia



Spout

- Lo spout inizialmente leggendo dal file metadata memorizza l’associazione id sensori e giocatore.
- Successivamente per ogni evento, controlla se questo appartiene ad un giocatore in campo, se avviene durante la partita ed infine inoltra usando il fieldsGrouping basato sull’id del giocatore la tupla al CellBolt.

Cell Bolt

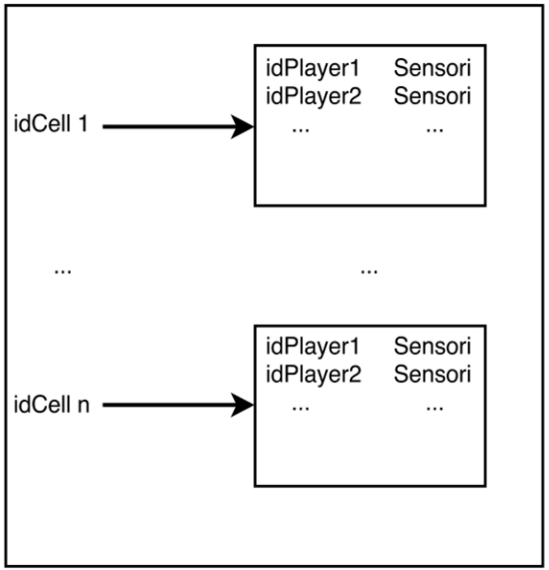
- Per ciascun giocatore calcola in quale cella delle 104 si trova in quel momento, valutando se questo ha effettuato uno spostamento o meno.
- Le tuple vengono emesse tramite fieldsGrouping basato sull'id della cella.
- Vengono inviati due tipi di tuple differenti al Percentage Bolt una contenente informazioni sullo spostamento, l'altra sull'evento stesso.

Percentage Bolt

Questo Bolt è in grado di gestire:

- L’arrivo di un evento di appartenenza ad una cella.
- L’arrivo di un evento di abbandono di una cella.
- L’emissione delle statistiche in base alle finestre temporali.
- L’arrivo tardivo di una cella ad un bolt.

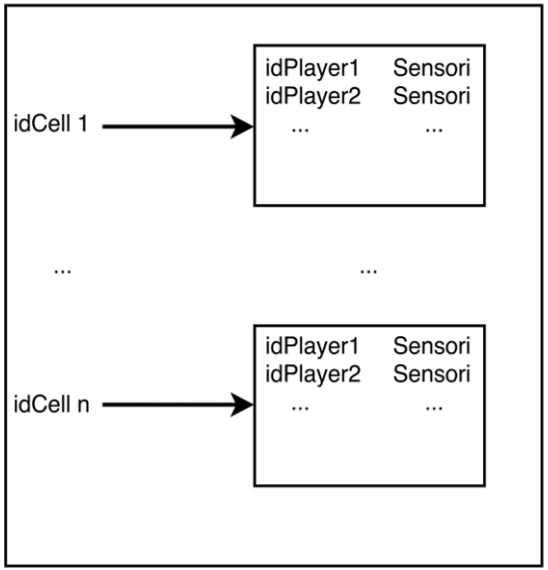
Percentage Bolt: strutture dati



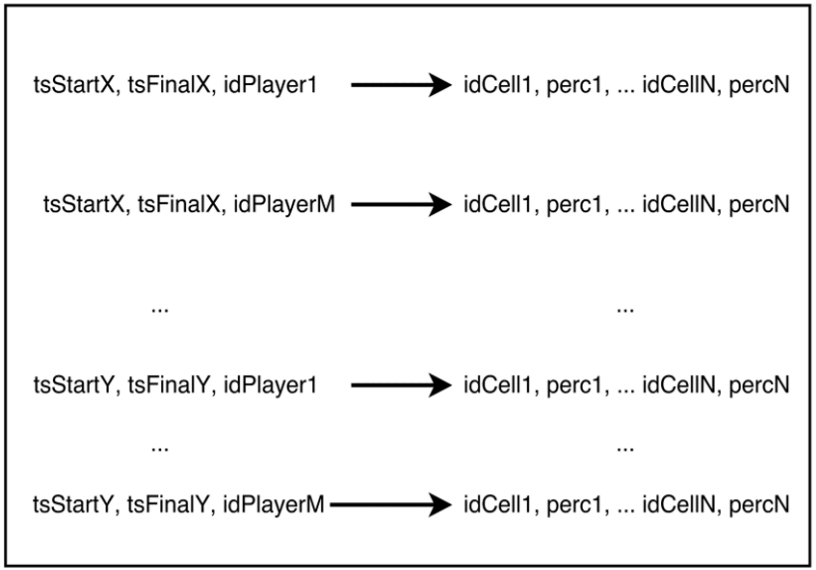
Dati memorizzati in una hashMap indicizzata sulle celle e con valore una hashMap contenente i dati dei sensori dei giocatori che sono stati in quella cella.

In fase di emissione si scorre questa hashMap e si preparano i dati da mandare al Merge Bolt.

Percentage Bolt: strutture dati



Dati dai quali estrarre le informazioni da inviare al MergeBolt



Dati da inviare al MergeBolt

Merge Bolt

Il Merge Bolt raccoglie per ogni giocatore le statistiche provenienti dalle varie celle in cui il giocatore ha militato ed emette l’output come richiesto dalla query.

Test

La macchina in cui sono stati effettuati i test è un HP 15-r127nl:

- Intel Core i7-4510U (2 GHz, 4 MB di cache, 2 core)
- 8 GB di SDRAM
- Sistema Operativo: Linux. (Kubuntu 16.04)

Si sono eseguiti diversi tipi di test che verranno riportati in formato tabulare e grafica.

Prestazioni al variare dell'intervallo di emissione

- Tempi di esecuzione sul full-game con grado di parallelizzazione a 2

	QueryOne	QueryTwo	QueryThree
intervallo di 0.5 secondi	138 secondi	140 secondi	617 secondi
intervallo di 15 secondi	93 secondi	97 secondi	124 secondi
intervallo di 60 secondi	88 secondi	92 secondi	114 secondi

Prestazioni al variare del grado di parallelizzazione

- Tempi di esecuzione sul full-game con statistiche emesse ogni 15 secondi

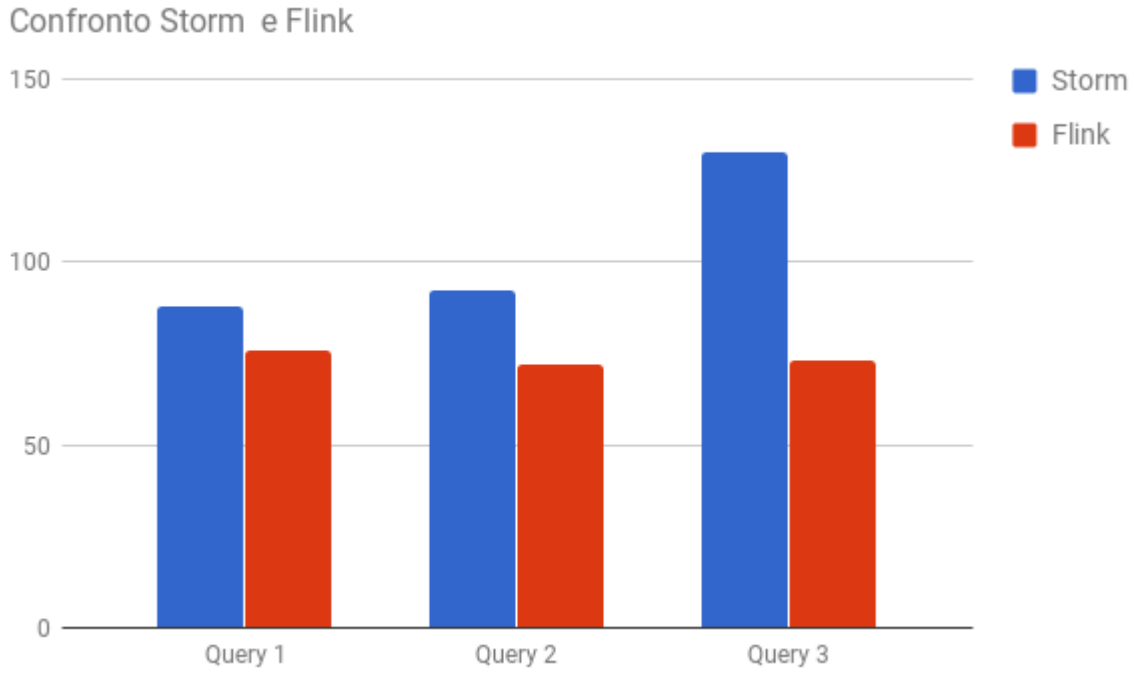
	QueryOne	QueryTwo	QueryThree
1 task per ogni bolt	95 secondi	101 secondi	118 secondi
2 task per ogni bolt	93 secondi	97 secondi	124 secondi
4 task per ogni bolt	105 secondi	111 secondi	147 secondi

Confronto di framework open-source di data stream processing

- Tempi di esecuzione sul full-game con statistiche emesse ogni 1 minuto, 2 task per ogni bolt.

	Storm	Flink
Query 1	88 secondi	76 secondi
Query 2	92 secondi	72 secondi
Query 3	114 secondi	73 secondi

Conclusioni e sviluppi futuri



STORM

Conclusioni e sviluppi futuri

- Data ingestion.
- Memorizzare in modo efficiente l’output.
- Valutare in termini di prestazioni la differenza tra Storm e Heron.
- Valutare i tempi di processamento su una piattaforma Cloud.

Gabriele Belli

GRAZIE PER L’ATTENZIONE.

Giuseppe Chiapparo