Università di Roma

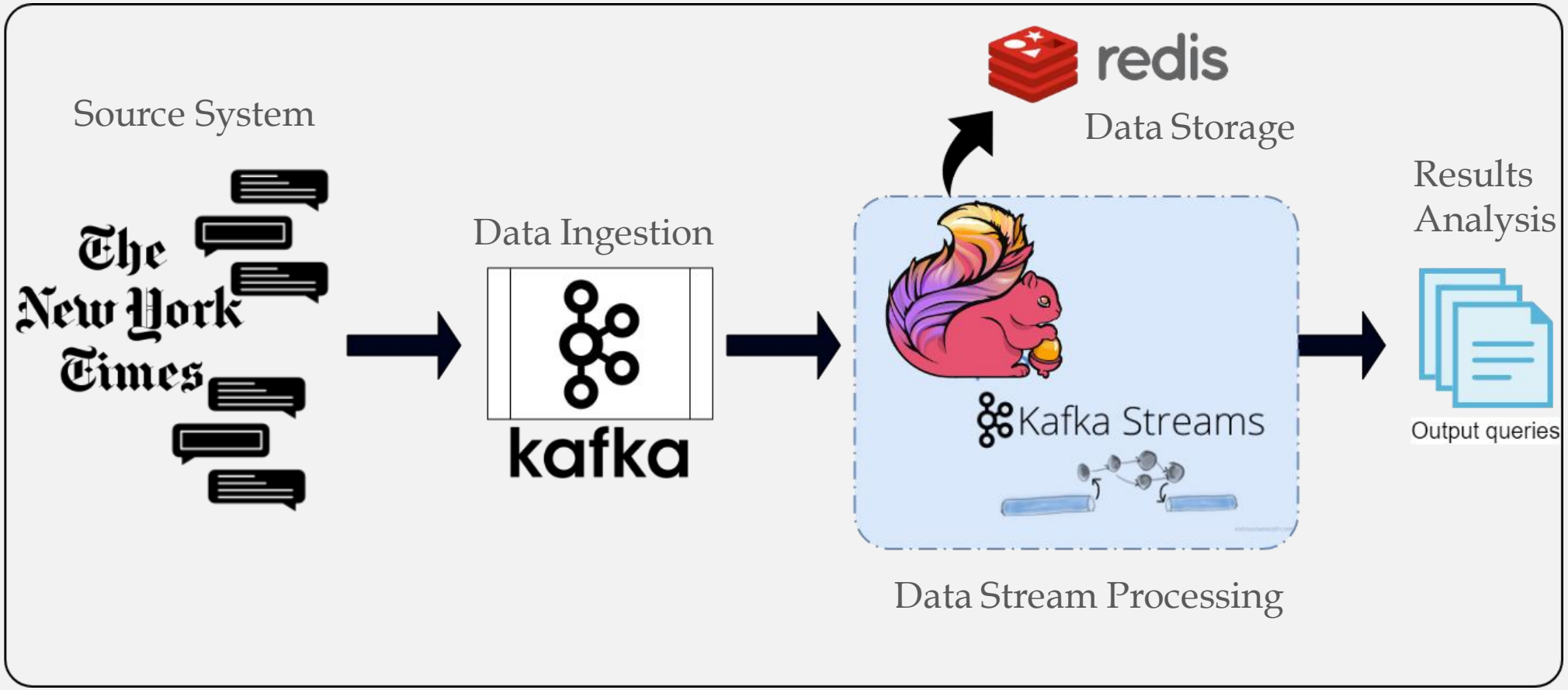
Tor Vergata

Analisi dei commenti di articoli pubblicati sul New York Times con Storm/Flink

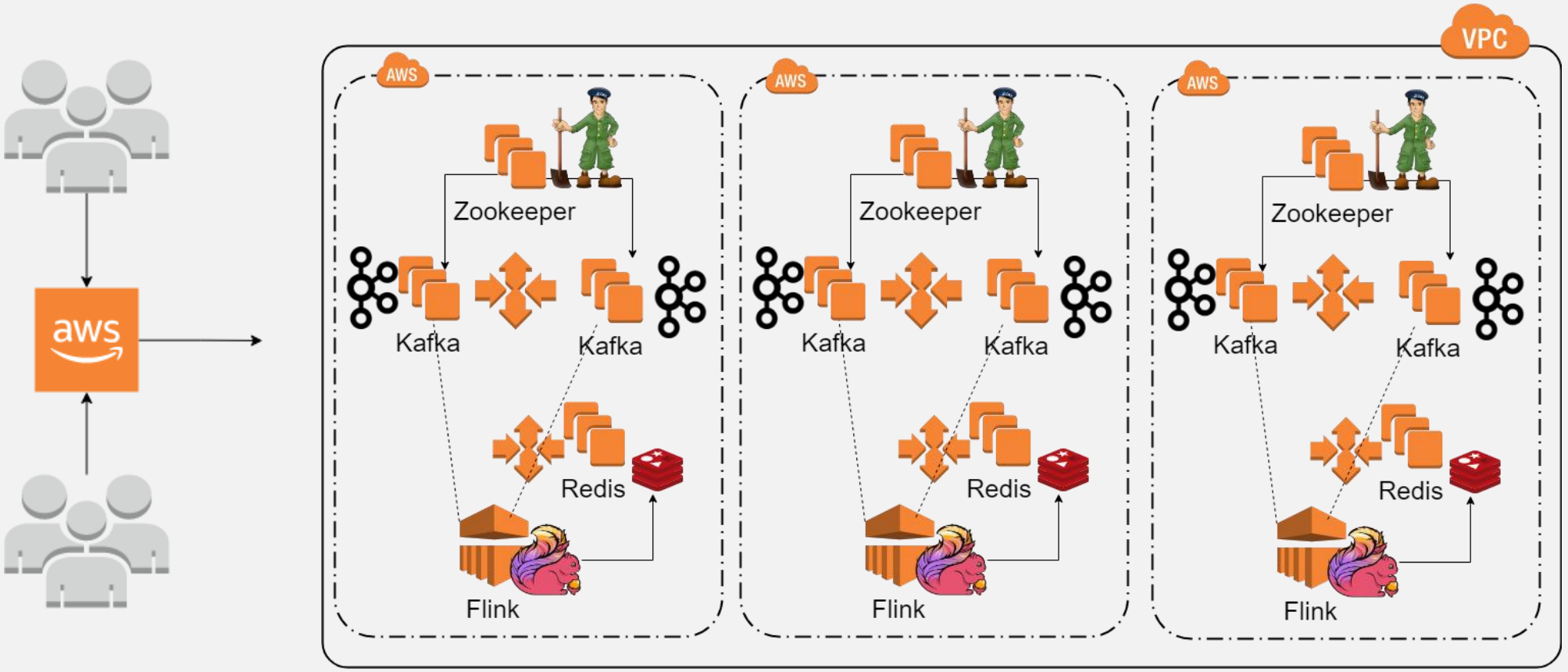
Sistemi e Architetture per Big Data - A.A. 2018/19

Di Cosmo Giuseppe- Nedia Salvatore

Architettura



Deploy


aws
VPC
AWS
Zookeeper
Kafka
Kafka
Redis
Flink
AWS
Zookeeper
Kafka
Kafka
Redis
Flink
AWS
Zookeeper
Kafka
Kafka
Redis
Flink

Gestione delle finestre

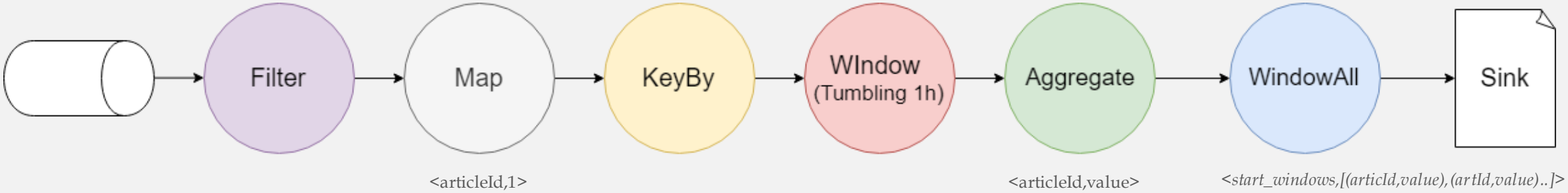
- **Flink**
  - Tumbling event windows
  - Finestra mensile customizzata
    - *startDate* = inizio del mese relativo alla tupla
    - *endDate* = fine del mese relativo alla tupla
- **KafkaStreams**
  - Sliding windows

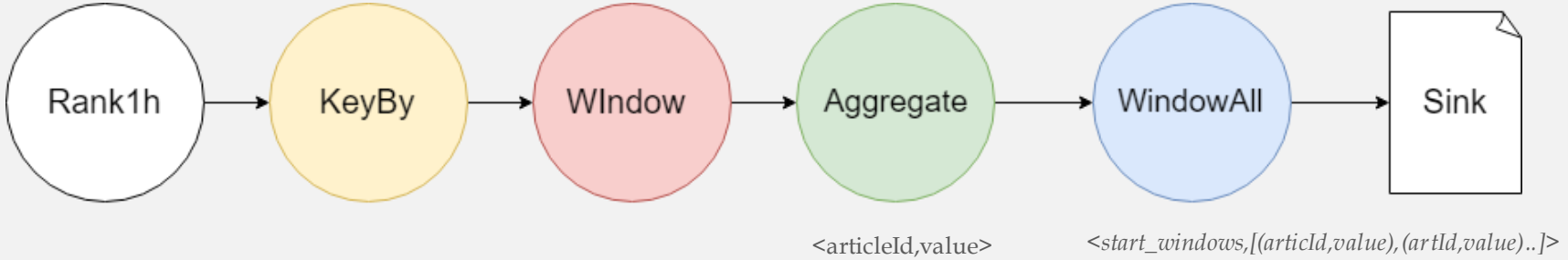
Ranking

- Obiettivo delle query 1 e 3
- Ottimizzazione attraverso l'uso di classifiche dinamiche
- Uso di Redis
  - Sorted set
    - **Key**: typewindow_query_startwindow
    - **Value**: id_value
    - **Score**: -value
    - Ogni chiave avrà un numero di elementi pari alle prime n posizioni(vengono scartate le tuple con posizione maggiore di quella prestabilita)
- Vantaggi
  - Carico estremamente ridotto per l'ultimo operatore
  - Delega dell' ordinamento a Redis
  - Riduzione di confronti

Queries

Query 1

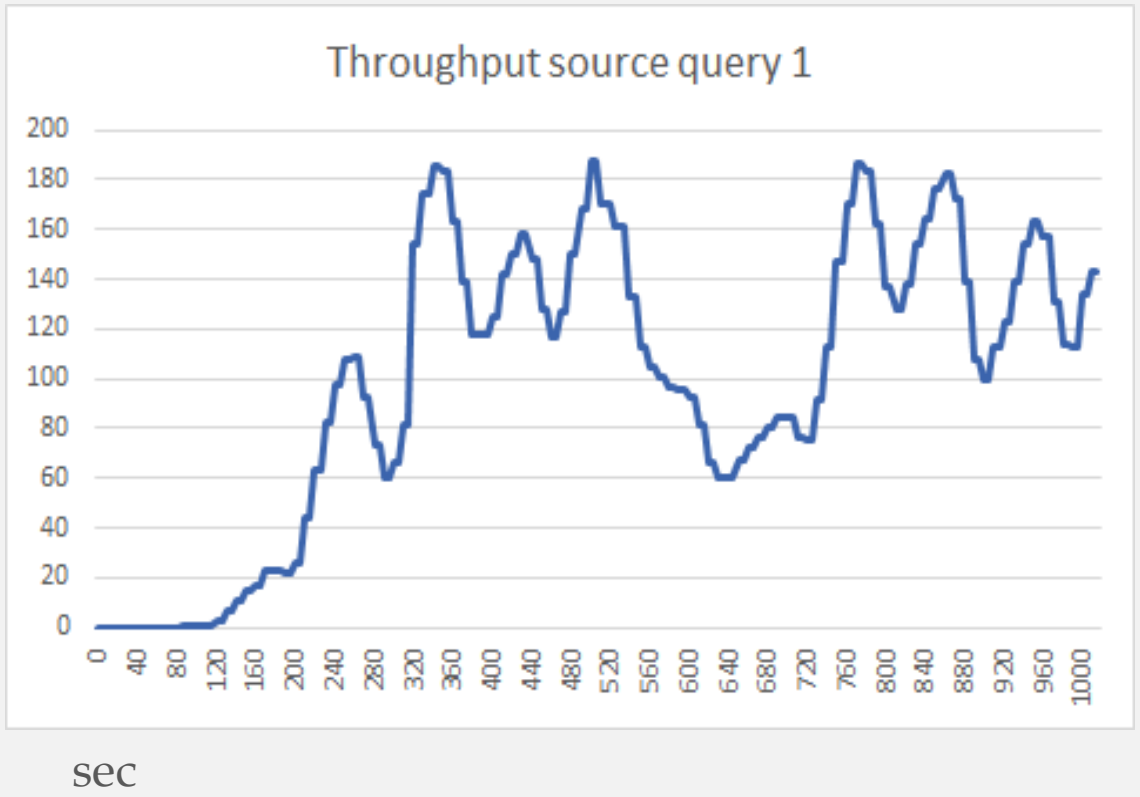

Filter
Map
KeyBy
WIndow
(Tumbling 1h)
Aggregate
WindowAll
Sink
<articleId,1>
<articleId,value>
<start_windows,[(articId,value),(artId,value)..]>

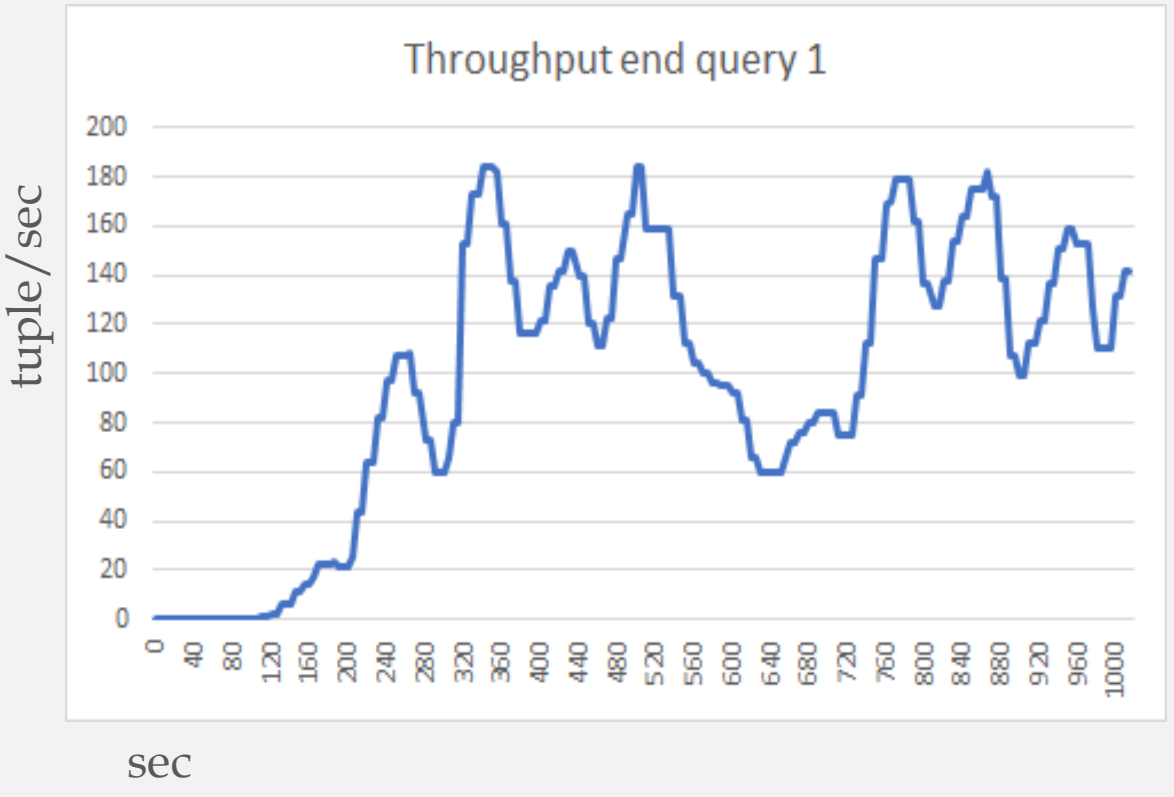

Rank1h
KeyBy
WIndow
Aggregate
WindowAll
Sink
<articleId,value>
<start_windows,[(articId,value),(artId,value)..]>

Analisi delle prestazioni di Flink

- Throughput: numRecordsOutPerSecond
- Cpu: System.CPU.Usage
- Latenza:
  - StartTime=istante di ingresso della tupla nel sistema
  - EndTime=istante in cui la tupla esce dall'operatore prima della finestra
  - Calcolata con classe Java **Instant**, essendo thread-safe a differenza di System.nanotime
- Tutte le metriche effettuate con un fattore di compressione di 1000(1 sec=1ms)

Performance Query 1: Throughput

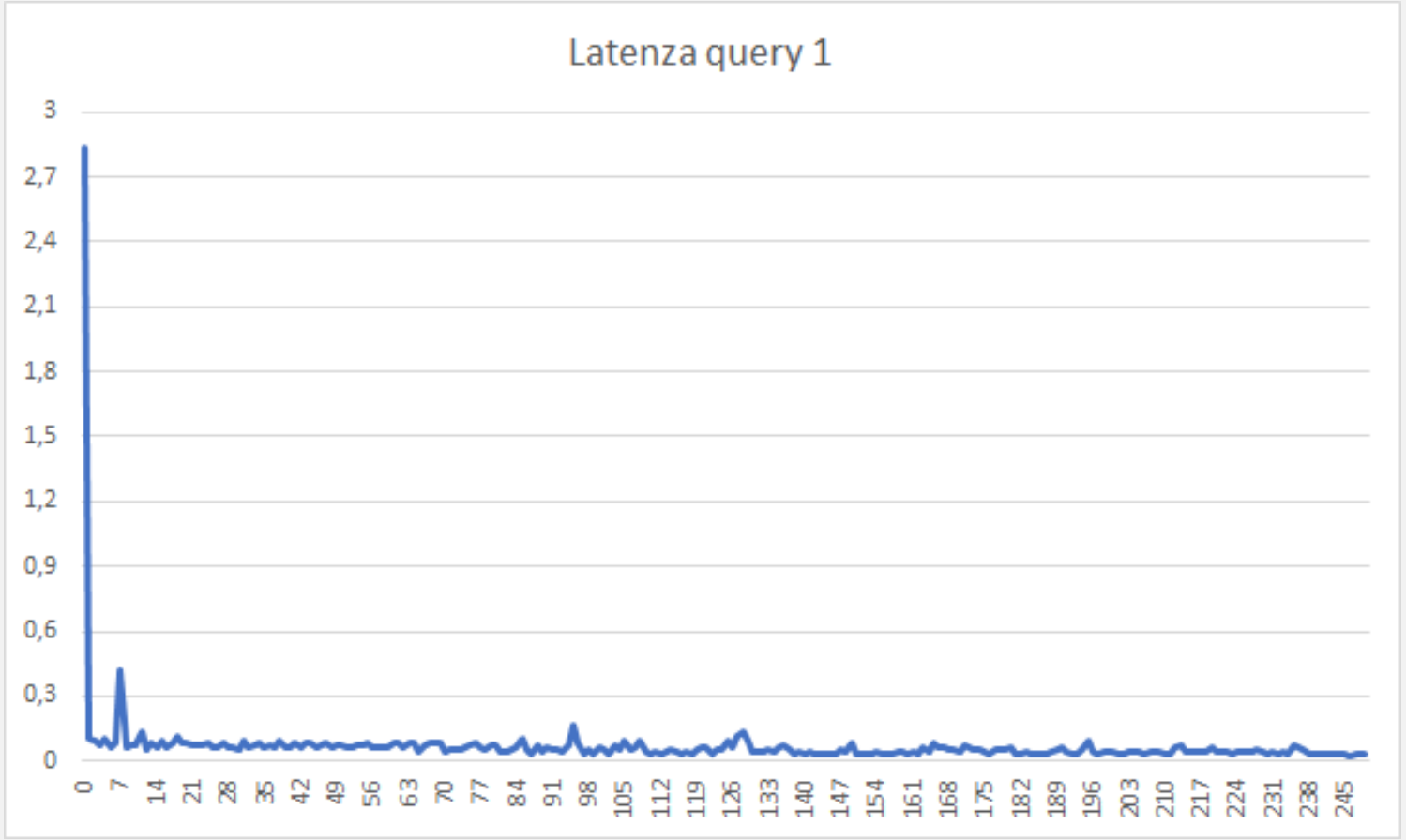




Fattore di compressione del simulatore: 1sec=1ms

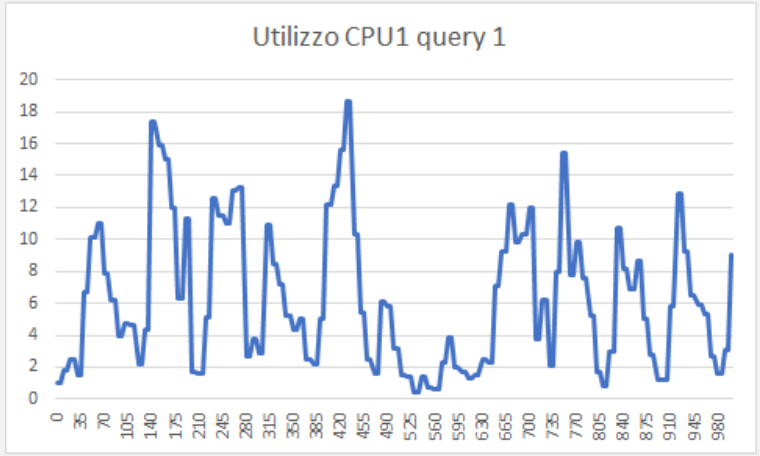
Performance Query 1: Latenza

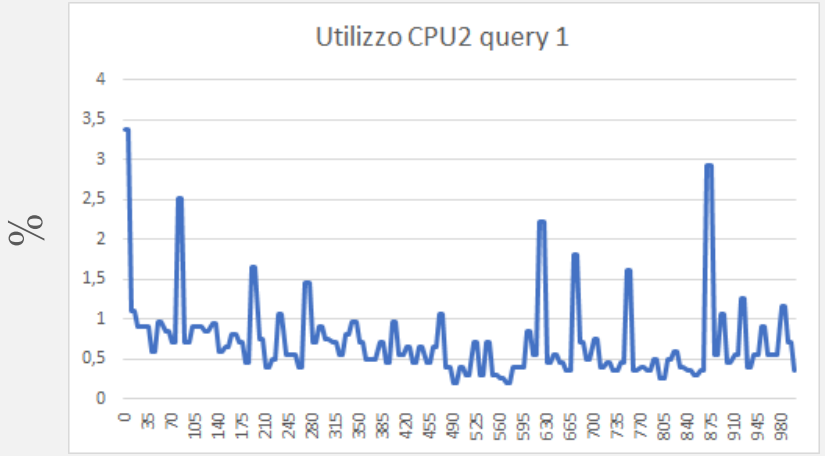
ms

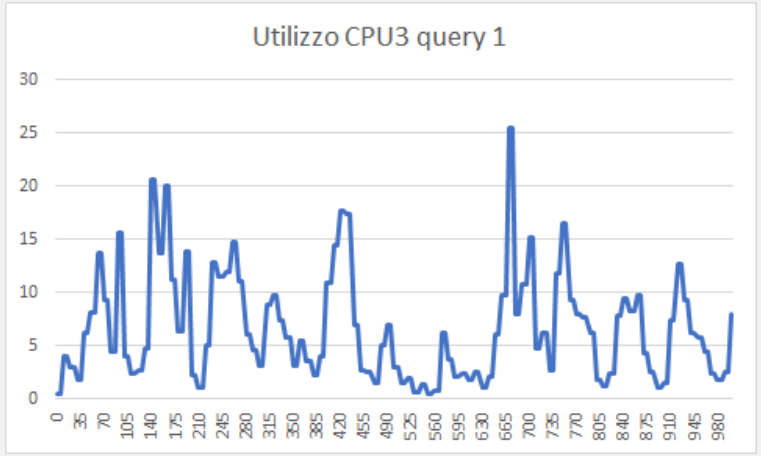


Index tuple

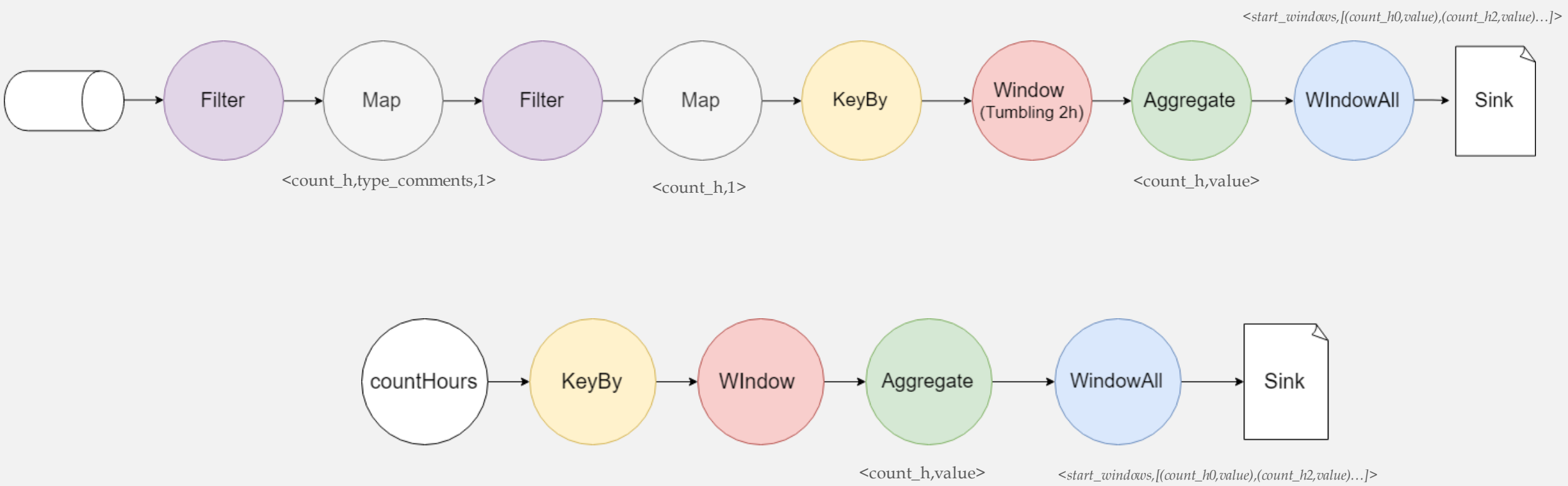
Performance Query 1: Utilizzo della CPU



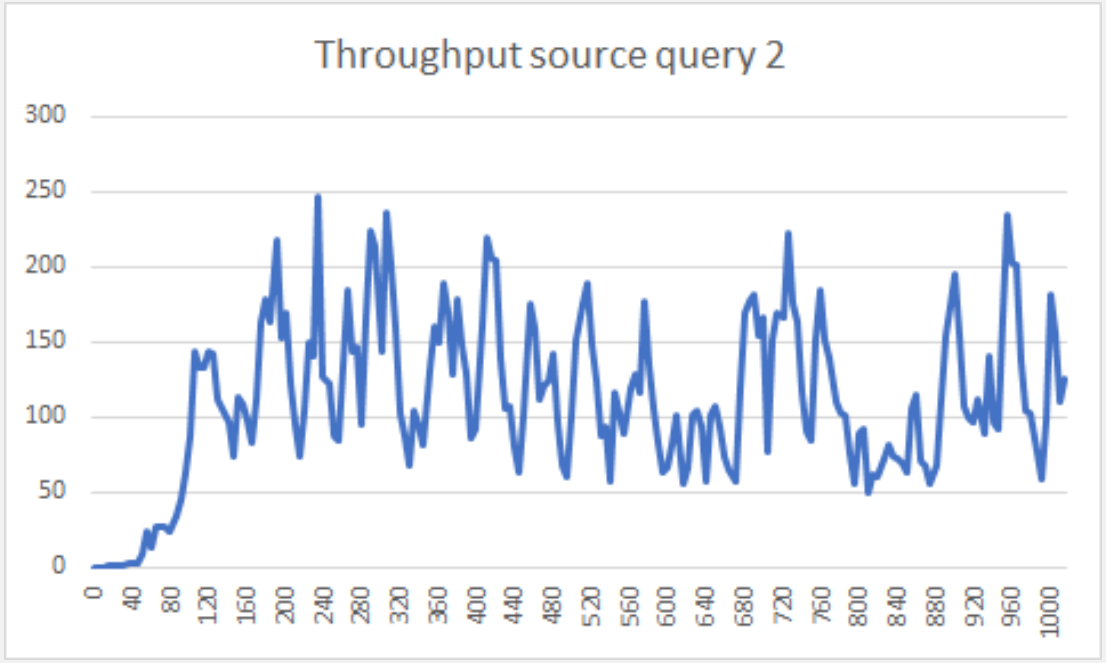


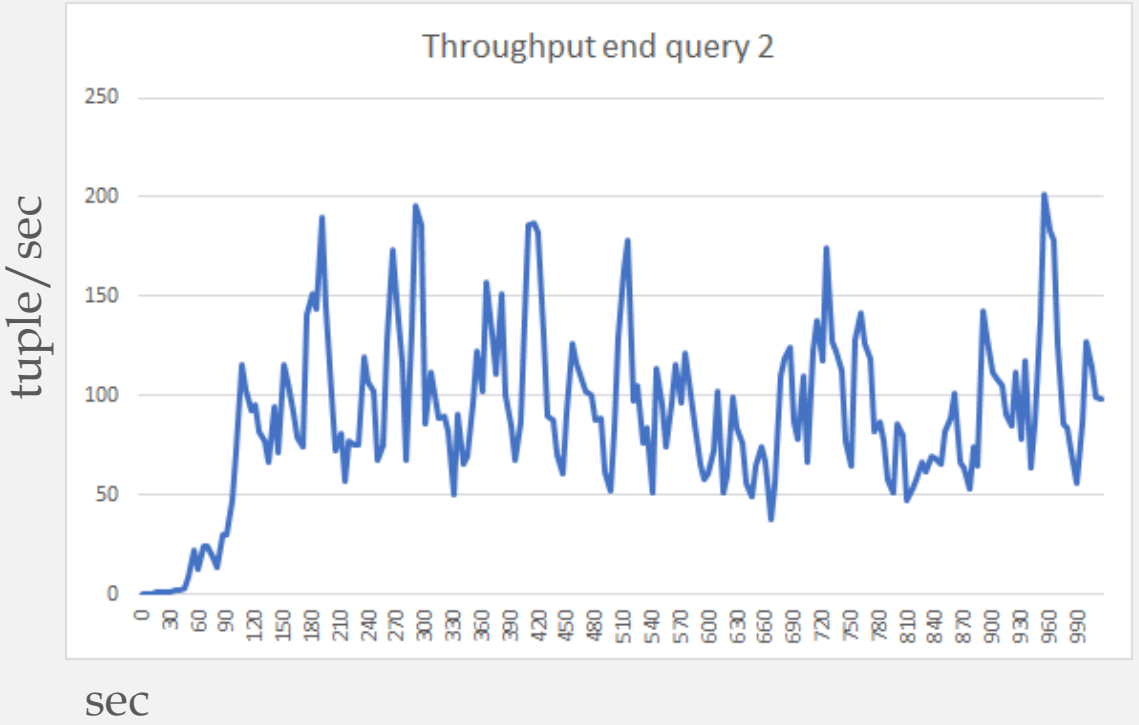


Query 2


<start_windows,[(count_h0,value),(count_h2,value)…]>
Filter
Map
Filter
Map
KeyBy
Window
(Tumbling 2h)
Aggregate
WIndowAll
Sink
<count_h,type_comments,1>
<count_h,1>
<count_h,value>
countHours
KeyBy
WIndow
Aggregate
WindowAll
Sink
<count_h,value>
<start_windows,[(count_h0,value),(count_h2,value)…]>

Performance Query 2: Throughput

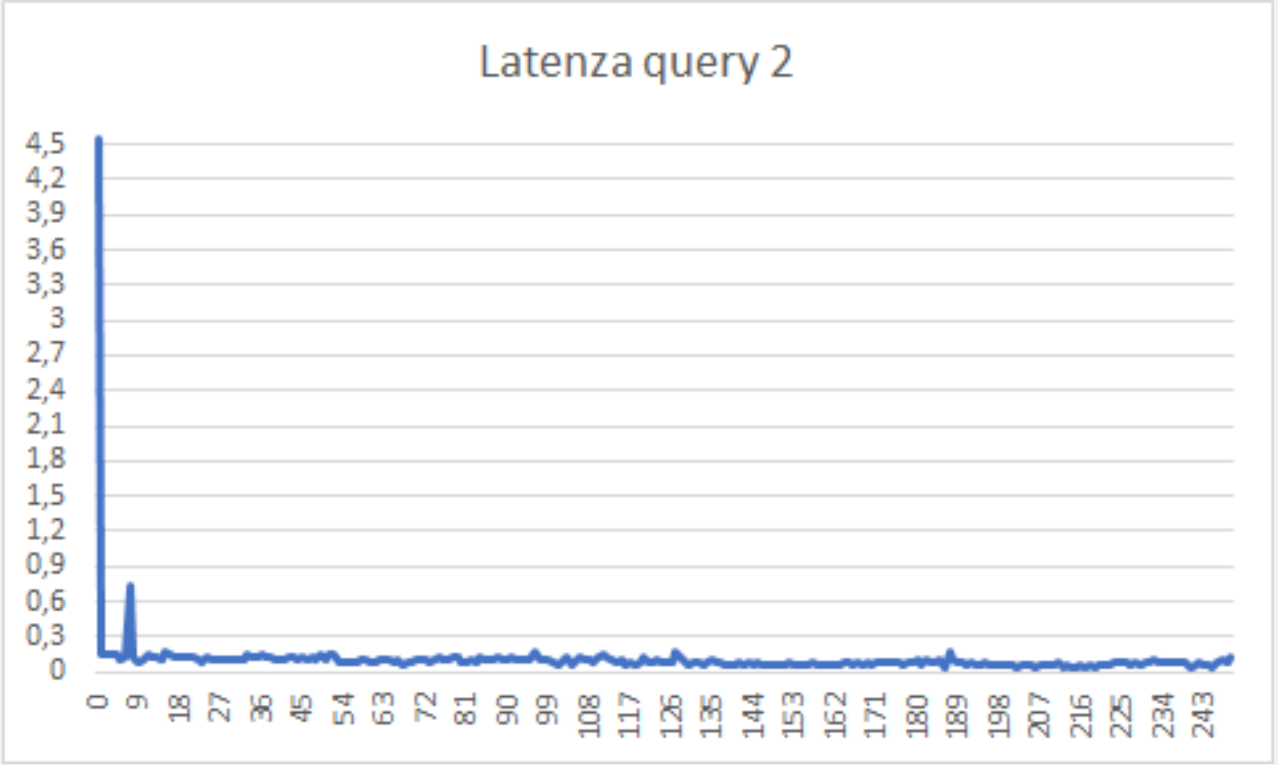




Fattore di compressione del simulatore: 1sec=1ms

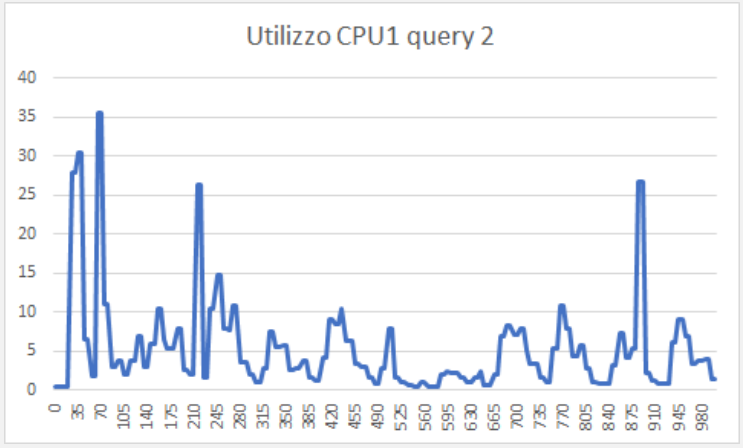
Performance Query 2: Latenza

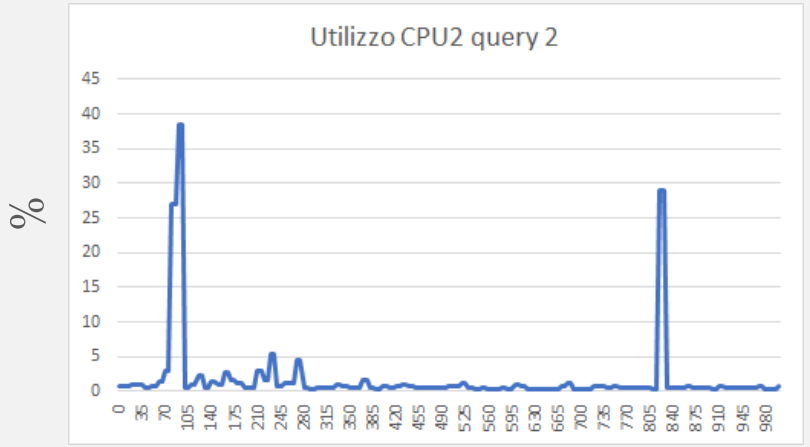
ms

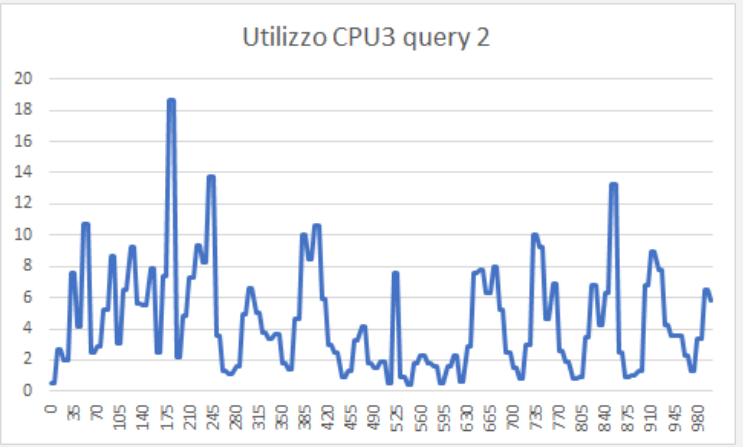


Index tuple

Performance Query 2: Utilizzo della CPU

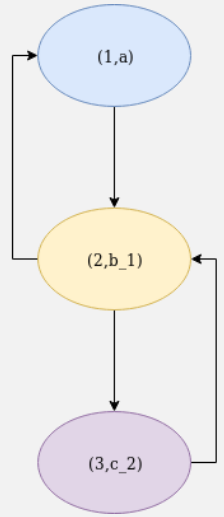




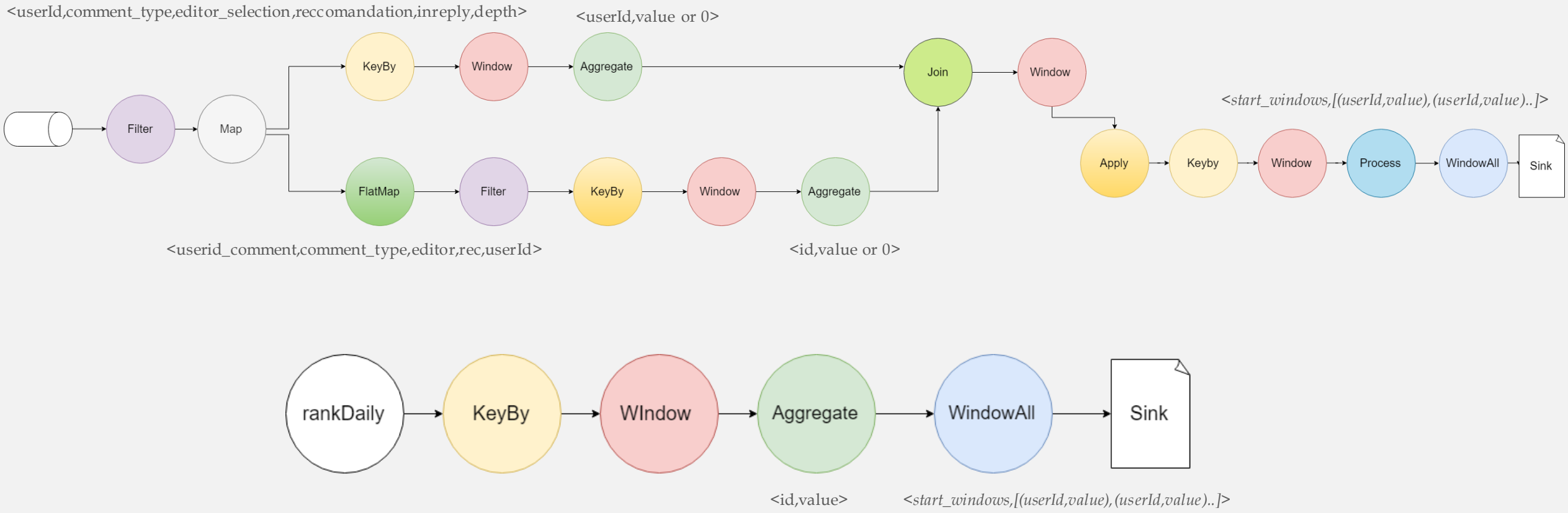


Query 3: Calcolo del grado di popolarità

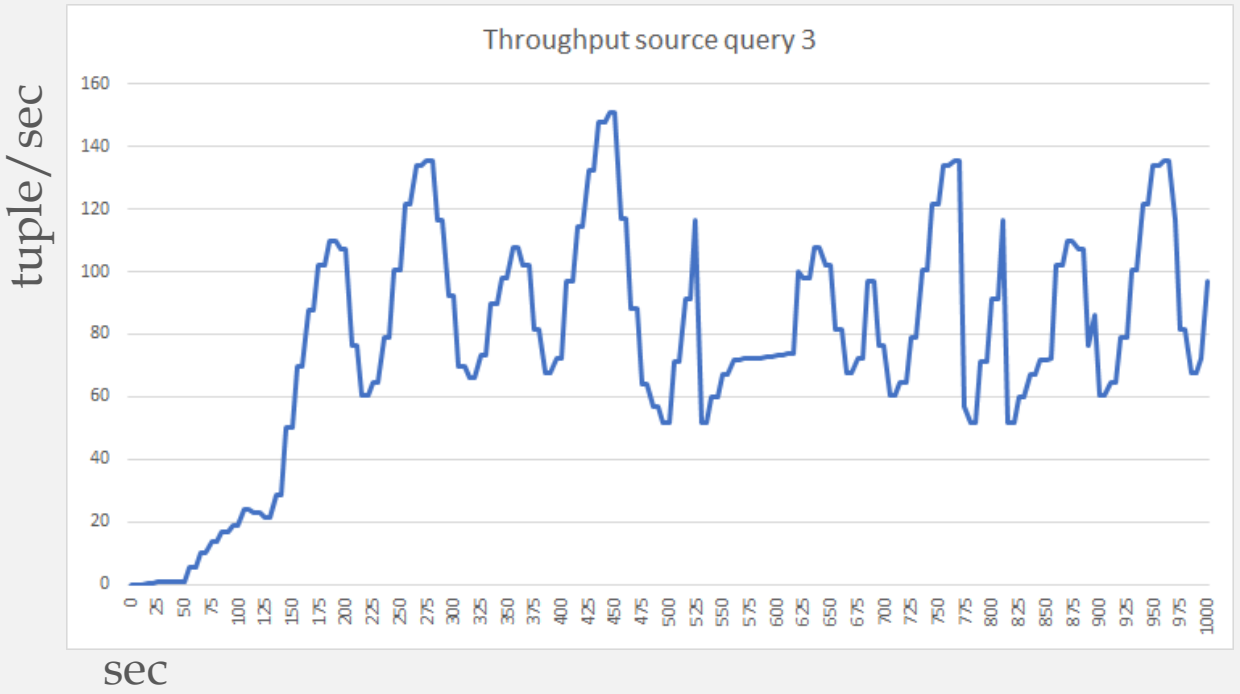
- Popolarity=wa*a+wb*b
- Calcolo di b
  - Necessario mantenere un mapping tra commentId e userId
  - Necessario mantenere la lineage della discussione
  - Uso di Redis
    - Key: commentId
    - Value: userId_inreplyto
  - Implementazione di Garbage Collector: tuple cancellate dopo che sono in memoria da 31 giorni(uso della funzione setex)

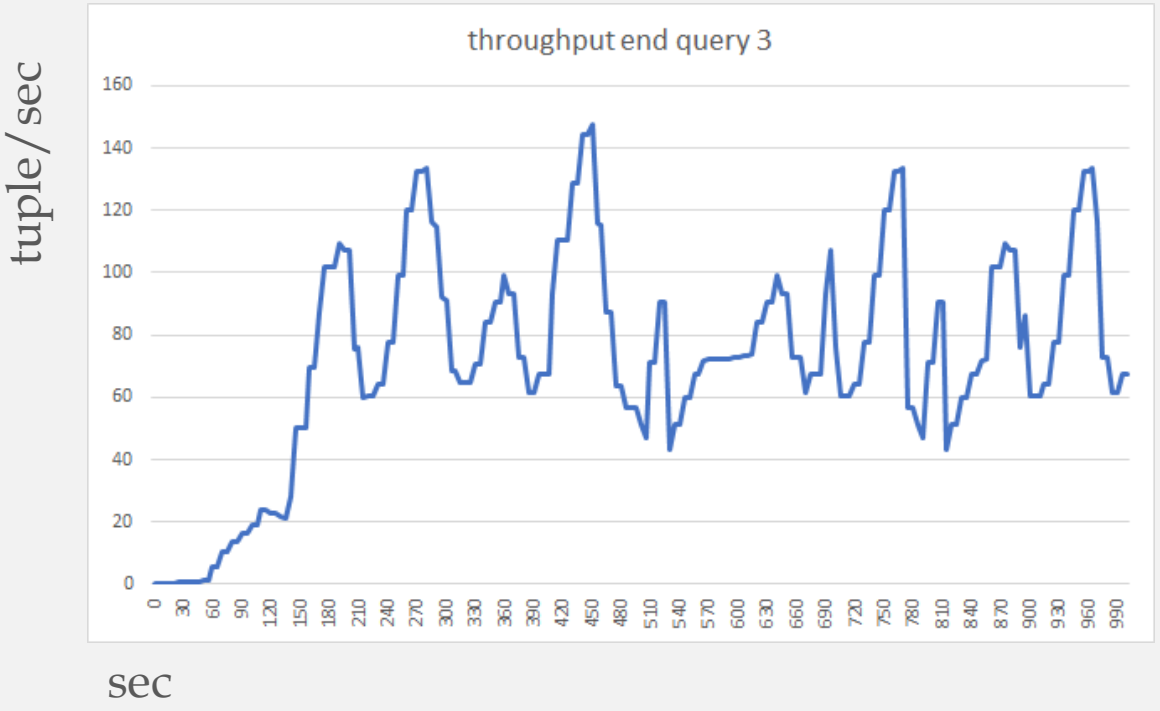


Query 3


<userId,comment_type,editor_selection,reccomandation,inreply,depth>
<userId,value or 0>
KeyBy
Window
Aggregate
Join
Window
Filter
Map
<start_windows,[(userId,value),(userId,value)..]>
Apply
Keyby
Window
Process
WindowAll
Sink
FlatMap
Filter
KeyBy
Window
Aggregate
<userid_comment,comment_type,editor,rec,userId>
<id,value or 0>
rankDaily
KeyBy
WIndow
Aggregate
WindowAll
Sink
<id,value>
<start_windows,[(userId,value),(userId,value)..]>

Performance Query 3: Throughput

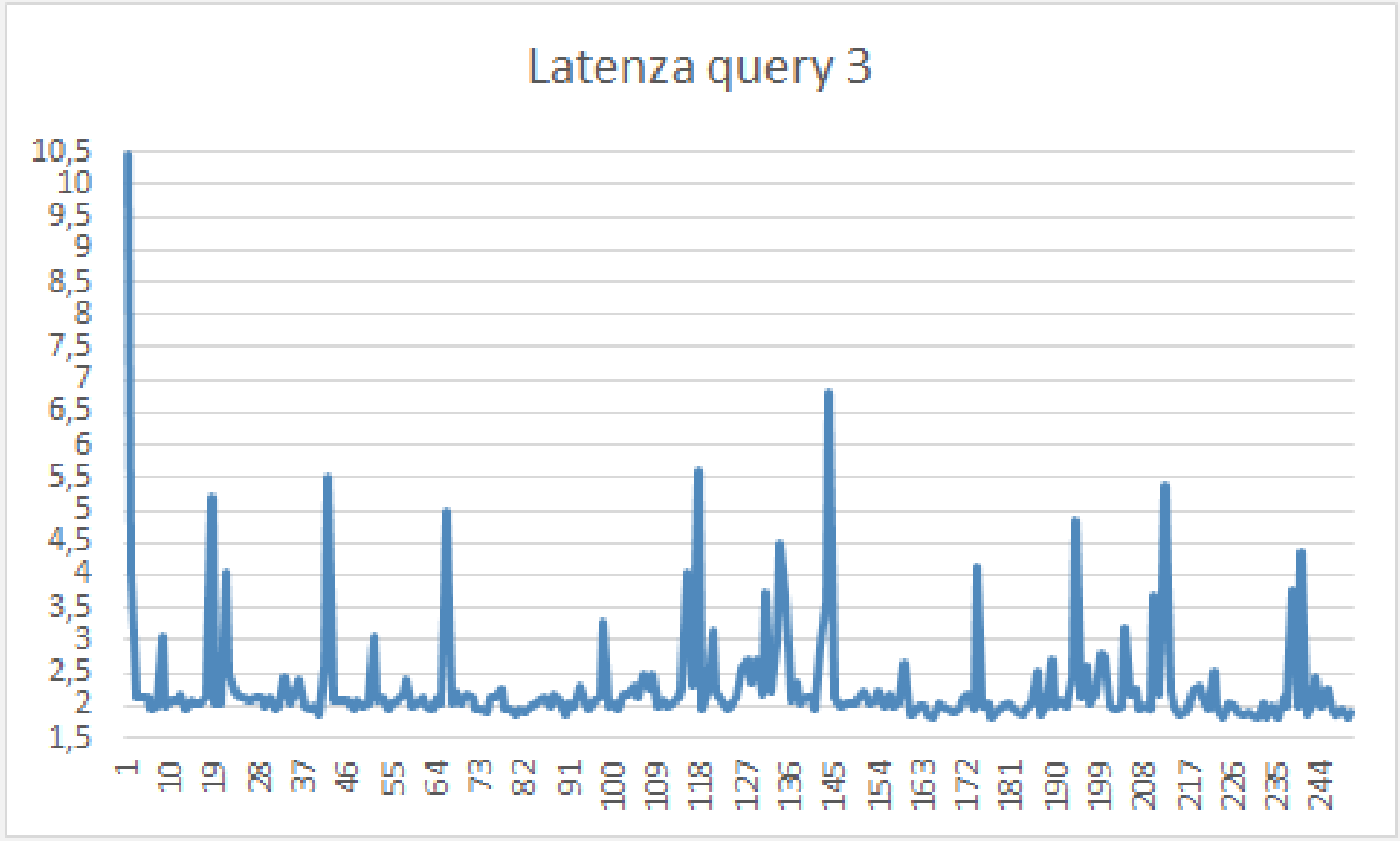




Fattore di compressione del simulatore: 1sec=1ms

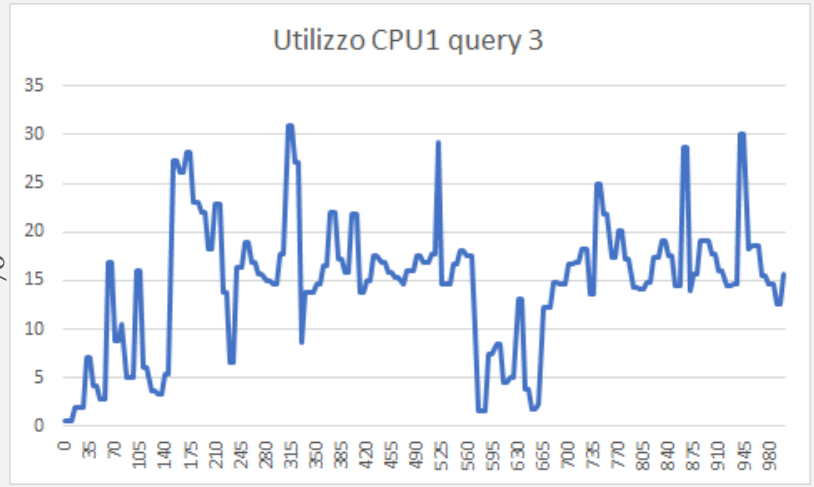
Performance Query 3: Latenza

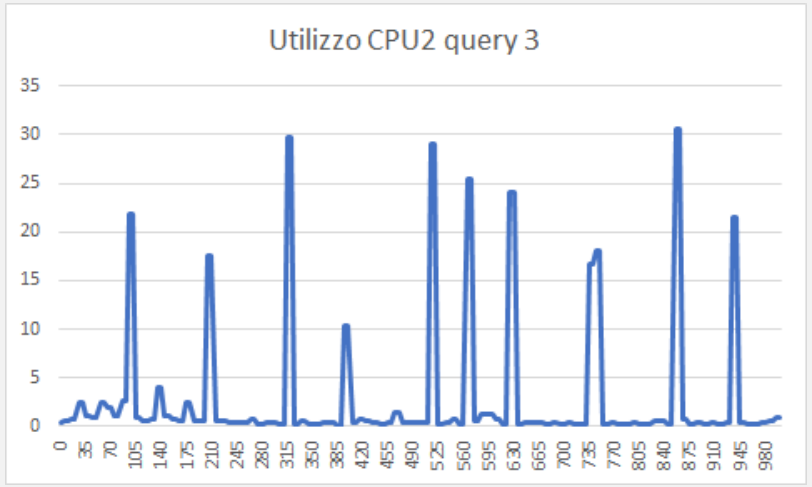
ms

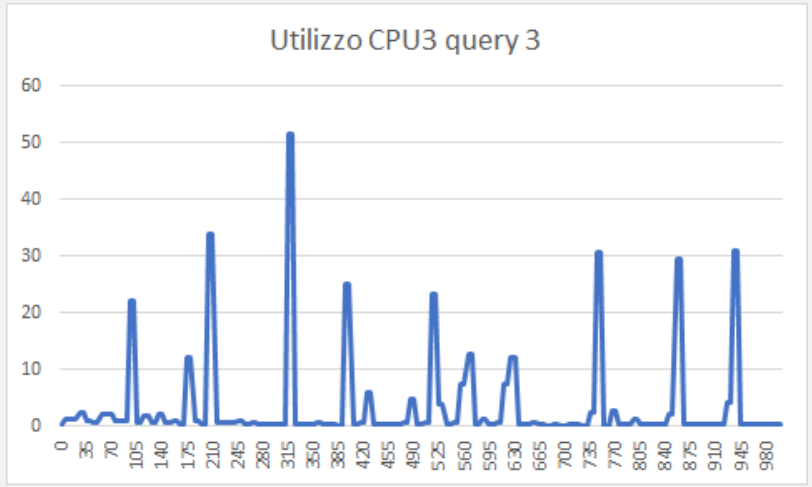


Index tuple

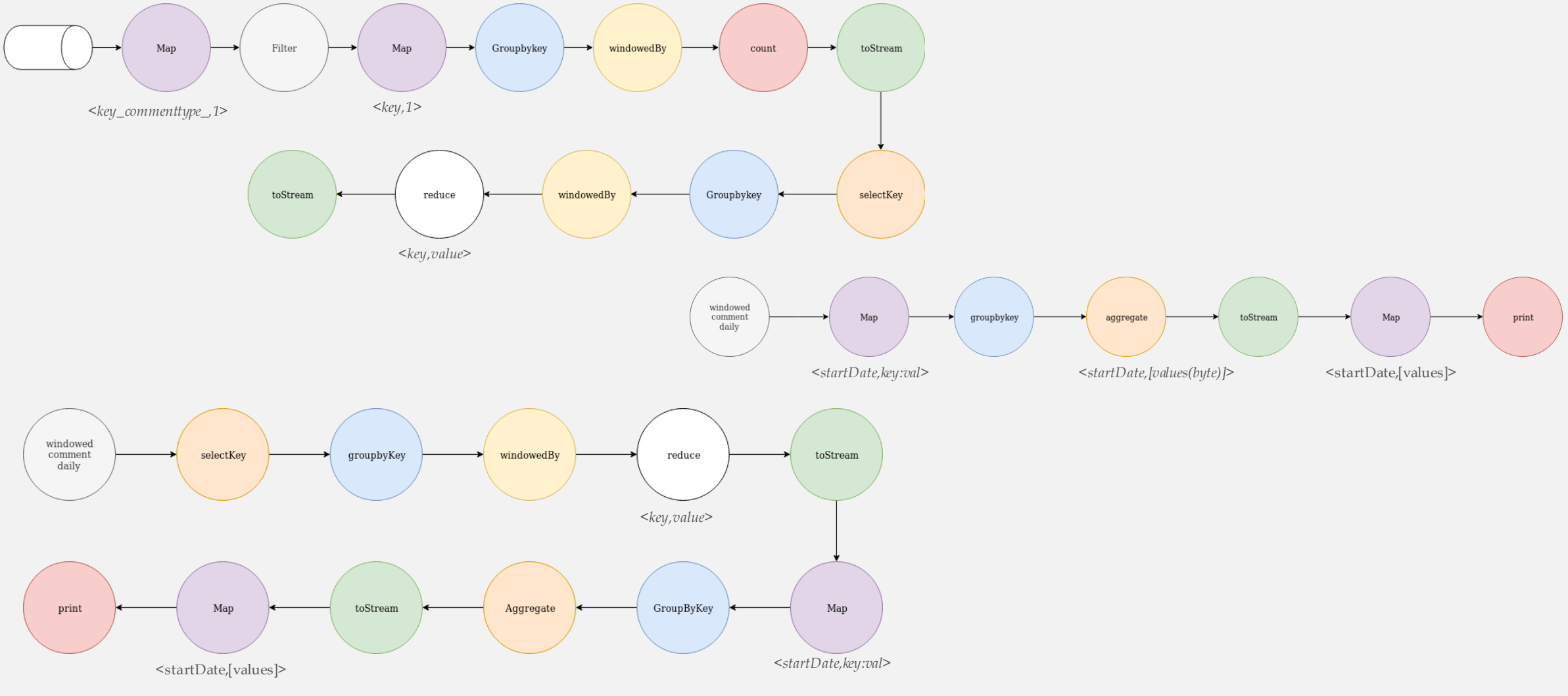
Performance Query 3: Utilizzo della CPU







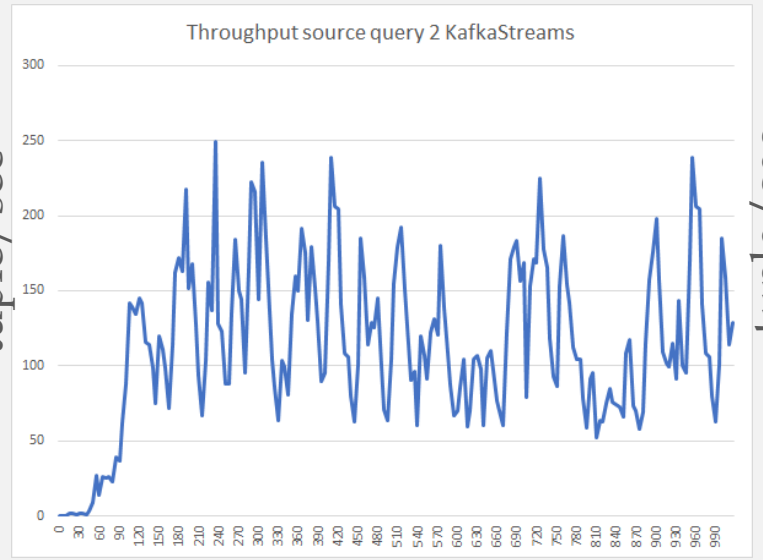
Query 2 KafkaStreams

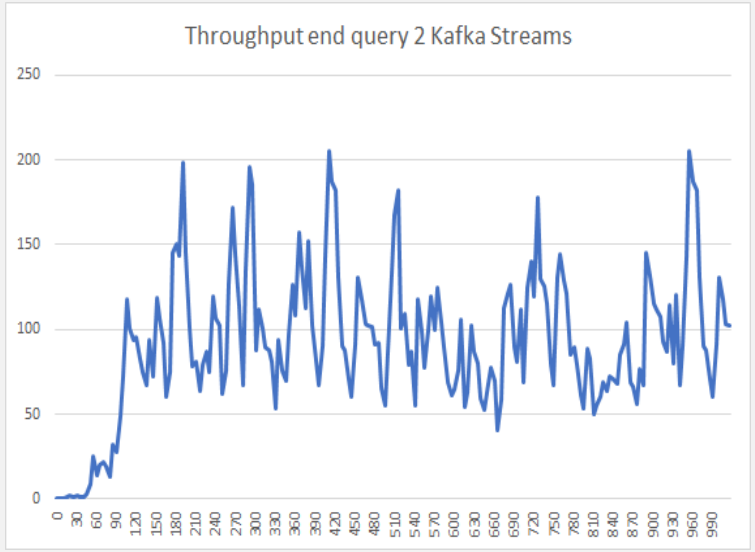


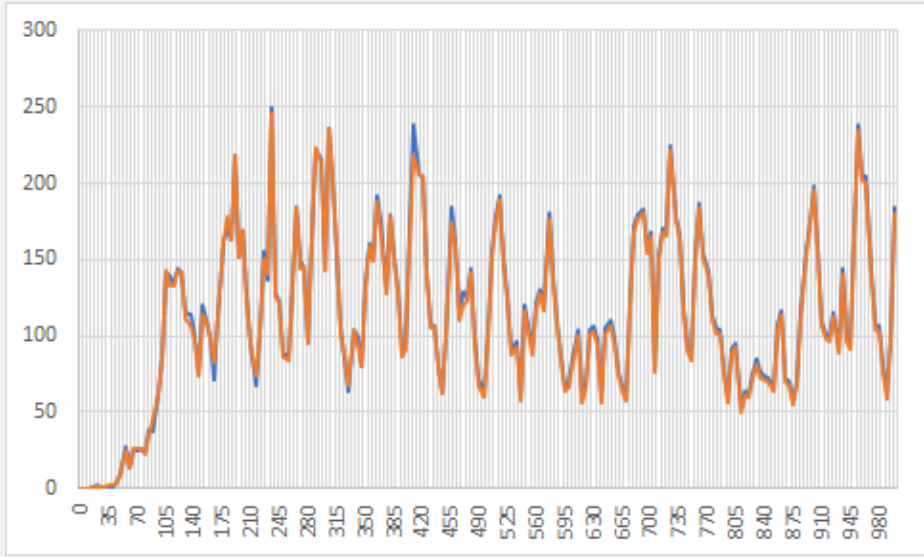
Analisi prestazioni Kafka Streams

- Throughput sorgente: forward-rate(primi 1000 sec)
- Throughput operatore: process-rate(primi 1000 sec)
- Latenza: process-latency(prime 250 tuple)
- Ottenute attraverso Jolokia
- Cpu: libreria Psutil Python e simpleHttp(primi 1000 sec)

Performance Query 2 Kafka Streams: Throughput



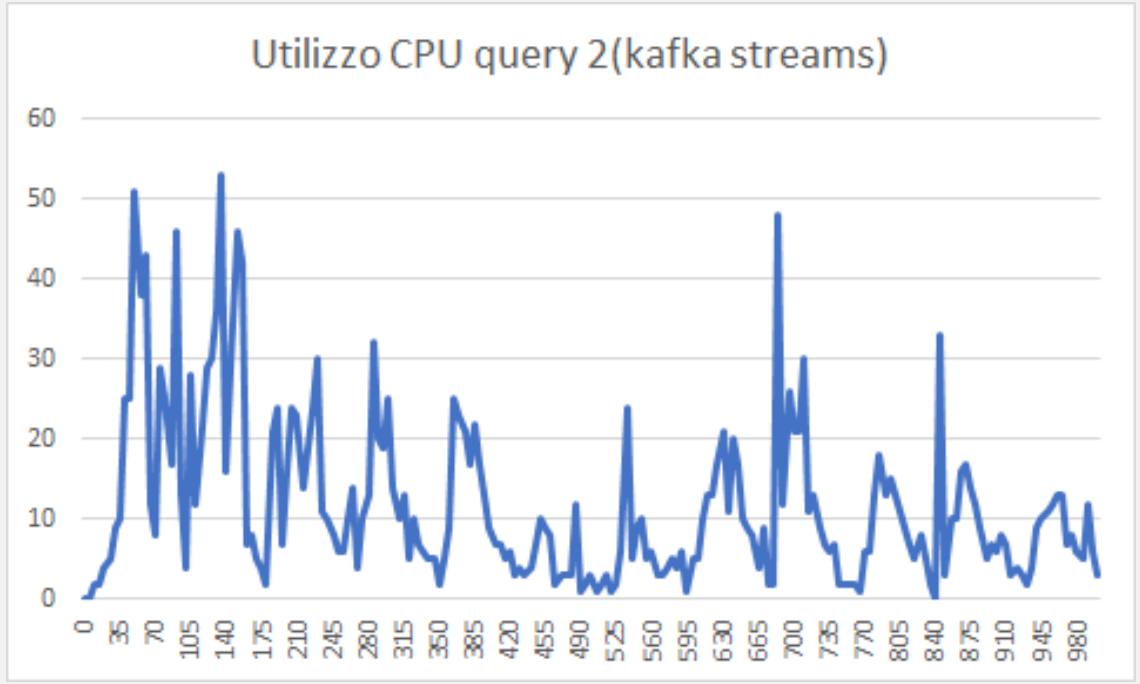


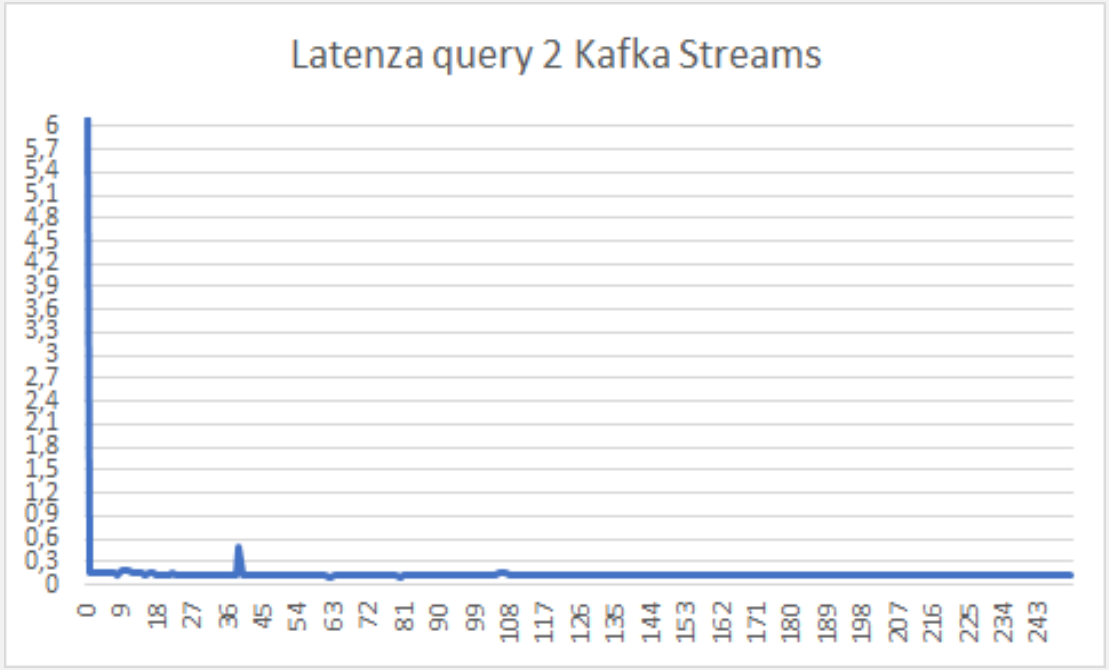


- Kafka Streams
- Flink

Fattore di compressione del simulatore: 1sec=1ms

Performance Query 2 Kafka Streams: Utilizzo CPU & Latenza





Grazie per l’attenzione!!!