Titolo: Confronto prestazionale tra supporti per il Data Stream Processing in contesti di applicazioni con stato “Larger-than-Memory”

Proposta

Il paradigma del Data Stream Processing studia l’elaborazione efficiente di computazioni che operano su sequenze infinite di dati trasmessi ad alta velocità (es. dati finanziari, reti di sensori, social media). Le elaborazioni compiute possono andare dal calcolo di statistiche (streaming analytics) ma anche elaborazioni più complesse come il processo di inferenza su modelli di ML/AI precedentemente addestrati oppure tecniche di learning continuo.

L’elaborazione di computazioni di streaming avviene mediante sistemi in cui le applicazioni sono sviluppate come grafi dataflow di operatori, ciascuno rappresentante uno stadio intermedio di trasformazione dei dati. Gli operatori computazionalmente più interessanti sono quelli con stato, solitamente operanti con stato partizionato. Ogni input dello stream è associato ad una chiave univoca, e la sua elaborazione prevede l’accesso in lettura e scrittura ad una ben determinata partizione dello stato associata a quella chiave. Nelle applicazioni più interessanti (analisi di social media, reti di sensori, analisi del traffico di rete) il numero di queste chiavi è molto elevato (anche milioni), e questo rendo lo stato complessivo più grande della memoria disponibile su un singolo calcolatore. Questo è sicuramente vero nel contesto in cui le applicazioni sono eseguite su dispositivi Edge/IoT con capacità di memoria di pochi GB, ma tale scenario può verificarsi anche su server (con centinaia di GB o qualche TB) in contesti più “spinti”.

Per gestire questa tipologia di stato, molti sistemi di streaming chiamati *Stream Processing Engine* (SPE) forniscono opportune API per sviluppare lo stato dell’operatore, e consentono di memorizzarlo su diversi backend in base ai parametri di configurazione dell’applicazione. Tra questi sistemi di backend per la memorizzazione persistente citiamo sistemi *Key-Value Store* quali RocksDB e Faster, che possono essere usati all’interno dei più diffusi SPE trasparentemente da parte del programmatore.

Strumenti

Il candidato prenderà dimestichezza con il linguaggio di programmazione C/C++, relativamente all’utilizzo della libreria WindFlow sviluppata dal Dipartimento di Informatica (un prototipale SPE per sistemi shared memory), e con i linguaggi Java e Scala per utilizzare SPE open-source più popolari ed industrialmente diffusi come Apache Flink.

Obiettivo

Lo studente dovrà implementare un certo numero di applicazioni di streaming in cui si utilizzi il supporto persistente per lo stato interno degli operatori. L’obiettivo è confrontare le prestazioni tra SPE diversi che forniscono supporto per operatori persistenti. Sicuramente lo studio verterà su WindFlow, che rappresenta una libreria di ricerca del Dipartimento di Informatica, ma dovrà essere studiato anche il comportamento di SPE indipendenti come, tra tutti, Apache Flink. Il goal finale è confrontare le prestazioni di SPE diversi sulle stesse applicazioni di benchmark, capirne le differenze e determinare possibili ottimizzazioni nel caso di WindFlow (se risultasse necessario).

Prerequisiti

- Architetture degli Elaboratori e Sistemi Operativi
- Ingegneria del Software
- Una buona base di programmazione (con i linguaggi C/C++, Java/Scala)