Real-Time Analytics in JVM

Really?

Sanjeev Kulkarni
PeerNova

Life of a tuple

Into a Queue -> Process -> No Longer needed

- Short-Lived Objects
- Object’s reclaimed in minor gc

Steady-State Characteristics

- Short JVM Heap Size

Complications

- Map-Side Aggregations
- Batching

JVM Heap Size: Small

Works best for steady state

Way too sensitive

- BackPressure in Heron triggered constantly

Throughput issues

- Map-Side Aggregation: Too much load on external systems
- Reduced Batching

JVM Heap Size: Big

Good for Throughput and External Services

GC Issues

- Objects start getting tenured
- Long GC cycles leads to unresponsive instances
- Different parts of topology get into this state at different times
- Topology is under constant backpressure

Result: The ‘Art’ of Tuning

Time Consuming

Strenuous

- Need to simulate different scenarios

Brittle

- Even minor traffic pattern changes needs a reconfig

Over Provision

- Wastage of resources

Good chance it won’t work even after all of this

Recommendations

Data Path has to be non JVM managed

Streaming solutions with Java interfaces ought to use something like unsafe for memory management

Current attempts

- Tungsten on Spark
- Unsafe in Presto/Flink

But leading Streaming Platforms still don’t use it

- Storm/Heron
- Samza