SDN based Cross-layer latency control for Data Stream Processing applications

Valeria Cardellini, Vincenzo Grassi, Francesco Lo Presti and Matteo Nardelli
University of Rome Tor Vergata, Italy
Introduction

IoT

Wide diffusion of sensing devices
Ubiquitous connection

Big Data

Volume (TB, PB)
Variety (text, multimedia)
Velocity (production, analysis)
Introduction

Big Data

Wide diffusion of sensing devices
Ubiquitous connection
Internet of Things

Volume (TB, PB)
Variety (text, multimedia)
Velocity (production, analysis)

Data Stream Processing (DSP)
continuously process streams of data generated by multiple, distributed sources, to extract valuable information

Existing Frameworks
• Apache Storm
• Twitter Heron
• Apache Flink
DSP Applications

DSP characteristics:

– Data Sources:
  • distributed
  • generate a \textit{continuous} stream of data (tuples)

– Stream: \textit{not stored}, directly fed to operators

– Operators (or Processing Elements):
  • Distributed
  • Work \textbf{in parallel}
  • Carry out a well defined operation
    (e.g., aggregation, filtering)
  • Usually produce an output stream
    which is fed to other operators
  • \textit{stateful}: output is function of the input and the internal state;
    \textit{stateless}: output is function of the input alone
The “Classic” Cloud Computing Approach

- Centralized computation at large-scale data centers
- “Distant” from data sources and information consumers
- But...
  - Is there enough bandwidth to collect the data?
  - Can (Big) data quickly move between computational nodes?
The Distributed Cloud Approach

To increase scalability and reduce latency, a possible solution is to rely on distributed and Fog-computing/near-edge computation.

Main idea: move computation “near” to data sources/destinations
Storm Architecture
Storm in a Distributed Environment

Official release of Storm

Centralized scheduler (Nimbus)

Round-robin scheduling policy

Agnostic to network delays and other nodes’ attributes

Not aware of environmental and stream rate changes

Our Extension of Storm

Distributed scheduler (Supervisor)

QoS-aware scheduling policy

Enhance the system with adaptation capabilities, introducing the MAPE model

poor performance in geographically distributed and highly dynamic environment
Distributed Scheduling in Storm
Operator Placement Problem

Scheduler: Component that assigns the operators to available computational resources.
Operator Placement Problem

• Operator $i$ characterized by
  - $C_i$ required computing resources
  - $R_i$ execution time per unit of data
  - $\lambda_{i,j}$ data rate between operator $i$ and $j$

• Resource node $u$ characterized by
  - $C_u$ available computing resources
  - $S_u$ processing speed
  - $A_u$ availability
  - $d_{u,v}$ network delay between node $u$ and $v$
  - $B_{u,v}$ available bandwidth between node $u$ and $v$

• Optimization Objective
  - Minimize end-to-end delay, reliability, network usage and/or a combination thereof
ILP Formulation

\[
\text{max } F'(x, y) \\
\text{subject to:}
\]

\[
R(x) \geq \sum_{i \in p} \sum_{u \in V^i_{res}} x_{i,u} \frac{R_i}{S_u} + \sum_{k=1}^{n_p-1} \sum_{(u,v) \in V^i_{res} \times V^j_{res}} y_{(i_k,i_{k+1}),(u,v)}d_{u,v} \quad \forall p \in \pi_c
\]

\[
\log A(x) = \sum_{i \in V_{dsp}} \sum_{u \in V^i_{res}} a_u x_{i,u} + \sum_{(i,j) \in E_{res}} \sum_{(u,v) \in V^u_{res} \times V^v_{res}} a_{u,v} y_{(i,j),(u,v)}
\]

\[
B_{u,v} \geq \sum_{(i,j) \in E_{dsp}} y_{(i,j),(u,v)} \lambda_{i,j} \quad \forall u \in V_{res}, v \in V_{res}
\]

\[
\sum_{i \in V_{dsp}} C_i x_{i,u} \leq C_u \quad \forall u \in V_{res}
\]

\[
\sum_{u \in V^i_{res}} x_{i,u} = 1 \quad \forall i \in V_{dsp}
\]

\[
x_{i,u} = \sum_{v \in V^j_{res}} y_{(i,j),(u,v)} \quad \forall (i,j) \in E_{dsp}, u \in V^i_{res}
\]

\[
x_{j,v} = \sum_{u \in V^i_{res}} y_{(i,j),(u,v)} \quad \forall (i,j) \in E_{dsp}, v \in V^j_{res}
\]

\[
x_{i,u} \in \{0,1\} \quad \forall i \in V_{dsp}, u \in V^i_{res}
\]

\[
y_{(i,j),(u,v)} \in \{0,1\} \quad \forall (i,j) \in E_{dsp}, (u,v) \in V^i_{res} \times V^j_{res}
\]

**Problem Variables:**
- \(x_{i,u} = 1\) operator \(i\) placed on node \(u\)
- \(y_{(i,j),(u,v)} = 1\) logical link \((i,j)\) mapped to network path \((u,v)\)

**Constraint Types:**
- **Response Time**
- **Availability**
- **Network Bandwidth and Node Capacity**
- **Assignment and Integer Constraints**
...other bits

• We expanded the framework to handle elasticity and runtime reconfiguration
  – Input rate change
    • Adjust operator replication by scaling out/in
  – Network delay variation
    • Operator Migration

• Issues addressed:
  – Migration of Stateful Operator
  – Cost of Reconfiguration
    • Application downtime impact system availability and performance
Elastic Scaling
Elastic Scaling
...other bits

• We expanded the framework to handle elasticity and runtime configuration
  – Input rate change
    • Adjust operator replication by scaling out/in
  – Network delay variation
    • Operator Migration

• Issues addressed:
  – Migration of Stateful Operator
  – Cost of Reconfiguration
    • Application downtime impact system availability and performance
Resource Control

...so far

• Complete Computational Resource Control
  – Cloud, Fog Nodes resources VMs, Docker Containers, etc.

• No Network Control
  – In a distributed environment network paths are determined by the operator underlying routing strategies
    • OSPF based network
Get the network in the loop

1. Legacy IP network
   - Non control over the network
     • OSPF Routing, operator controller
   - But we can measure end-to-end performance between endpoints

2. Hybrid IP-SDN Network (OSHI nodes)
   - SDN network monitor links performance metrics and exposes multiple “optimal” routes
     • According to different metrics, e.g., delay, hops, costs

3. Pure SDN Network
   - SDN controller exposes topology and links’ metrics
Operator Placement Problem
Legacy IP Network

QoS_{IPpath}

OP_i

OP_j
Legacy IP Network

- Distributed Storm
  - Keeps track of network QoS via own app level network path QoS monitoring
    - e.g. Vivaldi algorithm
  - Solves the Deployment and Run Time Reconfiguration based on network paths estimated QoS
  - Place operator as per optimization solution
Hybrid IP-SDN Network

- Use SDN to identify multiple paths between node pairs with different QoS attributes
Hybrid IP-SDN Network

- **SDN controller**
  - measures network QoS between hosts/nodes pairs
  - Identifies multiple QoS paths between node pairs
  - Info available via REST API
  - Setup chosen paths between endpoints

- **Distributed Storm**
  - Access QoS paths info via SDN interface
  - Solves the Deployment and Run Time Reconfiguration taking into account multiple paths (with different QoS) between nodes
  - Place operator as per optimization solution
  - Ask controller to setup chosen paths
SDN Network

- Use SDN to have complete network map and link delays
SDN Network

- **SDN controller**
  - Complete network topology map
  - measures per link QoS
  - Info available via REST API
  - Setup chosen paths between endpoints

- **Distributed Storm**
  - Access network map and links QoS via SDN interface
  - Solves the Deployment and Run Time Reconfiguration taking into account entire network topology and link QoS
  - Place Operator and ask for paths setup
Conclusions, not really!

- Optimal Deployment and Run-time Reconfiguration are ILP Problem
  - Do not scale, do they?
  - But they do provide a benchmark

- We are toying around with
  - Heuristics
  - Decentralized solutions
  - Multiple Application (topologies) at once
    - Game Theory approaches?
  - Other frameworks
  - Improve migrations support
    - Extremely costly operation!
  - Do SDN controllers scales?
  - ....
Thanks for your attention!

Lo.Presti@ing.uniroma2.it