Elastic deployment and adaptation for distributed stream processing applications

Valeria Cardellini
University of Rome Tor Vergata, Italy

ACROSS Final Workshop
Amsterdam, November 10th, 2017
Data stream processing (DSP)

- Process unbounded data streams generated by multiple, distributed sources
- Extract valuable information
- In a (near) real-time and reliable manner
Where to process data?

• To increase scalability and reduce latency

Exploit distributed and near-edge computation

Main idea: move computation close to data sources/destinations
New scenario with “old” and “new” challenges

• Stream data rates can be high (velocity) and data arrive in large volumes (volume)
• Processing stream data with QoS requirements
  – E.g., end-to-end latency
• Network latencies are significant
• Computing and networking resources are heterogeneous
• Data cannot be processed everywhere
• DSP frameworks are not designed for fog/edge distributed environments
Where to deploy applications?

**Operator deployment/placement**: Determine the nodes that will execute each operator of a DSP application.
ODP: Optimal DSP Placement

• We proposed ODP
  – Centralized policy for optimal placement of DSP applications
  – Formulated as Integer Linear Programming (ILP) problem

• Our goals:
  – Compute the optimal placement
  – Provide a unified general formulation of the placement problem for DSP applications (but not only!)
  – Consider multiple QoS attributes of applications and resources
  – Provide a benchmark for heuristics

ODRP: Optimal DSP Replication and Placement

• Exploit *data parallelism*: concurrent execution of multiple operator replicas on different data portions
ODRP: Optimal DSP Replication and Placement

Self-adapt at run-time

• Many factors may change at run-time
  – Load variations, QoS attributes of resources, cost of resources, network characteristics, node mobility, ...

• How to adapt the placement and replication when changes occur?

• What adaptation actions?
  – Migrate the operators
  – Elastically scale-out/in the number of operator instances
Runtime adaptation challenges

• Costs to reconfigure at run-time the deployment
• Negative impact on the application performance in the short term
  – Application freezing times caused by operator migration and scaling

Perform adaptation only when really beneficial

Take into account the overhead for migrating and scaling the operators (especially stateful ones)
Adaptation issues with stateful operators

• **Stateful** operators
  – Keep some sort of state across different tuples
  – E.g., counter

• Require mechanisms to
  – Partition streams and load balance among replicas
    • Support only partitioned stateful operators
  – Relocate the operator internal state on a different node
Stateful operator migration

- Pause-and-resume approach

Application latency peak during migration

1. Stop migrating operator
2. Save state
3. Terminate migrating operator and start it on new node
4. Restore state
5. Resume stream processing
Elastic stateful migration in Storm

• (Almost) No support by most popular DSP frameworks

• We developed Elastic Storm

• Relocate the operator internal state on a different node and enable Storm to change the application deployment at run-time
EDRP: Elastic DSP Replication and Placement

- Extend ODRP
- Provide a **general formulation**
- Consider multiple **QoS attributes** of applications and resources
- Model **adaptation costs** for migrating and scaling the operators
- Perform reconfigurations **only when needed**
- A **benchmark** for heuristics

EDRP: Elastic DSP Replication and Placement

\[
\min_{x, y, r, t_D} F'(x, y, r, t_D)
\]

subject to:

\[
\begin{align*}
    r & \geq R_\pi(x, y) \\
t_D & \geq T_{D, i}(x) \\
R(x, y) & \leq R_{\text{max}} \\
C(x, y) & \leq C_{\text{max}} \\
T_D(x) & \leq T_{D, \text{max}}
\end{align*}
\]

\[
Res_u \geq \sum_{i \in V_{\text{dsp}}} \sum_{u \in \mathcal{P}(V_{\text{res}}^i)} u_{Res} x_{i, u}
\]

\[
1 = \sum_{u \in \mathcal{P}(V_{\text{res}}^i)} x_{i, u}
\]

\[
x_{i, u} = \sum_{y(i, j) \in E_{\text{dsp}}} \sum_{u \in \mathcal{P}(V_{\text{res}}^j)} y(i, j, (u, v)}
\]

\[
x_{j, v} = \sum_{u \in \mathcal{P}(V_{\text{res}}^j)} y(i, j, (u, v)}
\]

\[
x_{i, u} \in \{0, 1\}
\]

\[
y(i, j, (u, v)} \in \{0, 1\}
\]

ACROSS Final Workshop
Elasticity capabilities in Storm

• Augmented Storm with MAPE capabilities and centralized placement
EDRP: Elastic DSP Replication and Placement

- Query 1 of DEBS’15 Grand Challenge: popular taxis route in NYC

### Source data rate (tuples/s)

![Source data rate graph]

### Response time (ms)

![Response time graph]

### Total number of replicas

![Total number of replicas graph]

**Minimization of response time and cost**

- **(without reconfiguration costs)**
- **ACROSS Final Workshop**
- **(with reconfiguration costs)**
Conclusions

• With EDRP we address some issues to achieve elastic distributed data stream processing in fog/edge computing environments
  – Place operators
  – Migrate and scale out/in operators
  – Manage stateful operators
Limitations

• Scalability of centralized optimization algorithm

• Scalability of centralized MAPE architecture
  – Distributed components but logic still centralized

Addressed in Francesco’s talk