Data Analytics at the Network Edge

Apostolos Papageorgiou
NEC Laboratories Europe
Heidelberg, Germany

Apostolos.Papageorgiou@neclab.eu

EU IoT Week, Belgrade, 31/6 - 2/7 2016
Outline

Background about Network-edge computing
  • Technical Landscape and motivation
  • Current limitations

Real-time per-item data reduction
  • Differentiators and overview of our solution
  • Way of operation of „exchangeable data handlers“
  • „Streamification“ of data reduction algorithms
  • Summary of evaluation

Edge deployment of IoT data streaming tasks
  • Stream Processing Frameworks and their limitations
  • Our solution for edge-aware streaming task deployment
Background

Technical landscape, motivation, and current limitations for Network-edge computing
Network-edge computing

Data Center, (Cloud)

Network Core

Multi-service Edge (Gateways, Edge routers)

Embedded Systems & Sensors (M2M devices)

Potentially monitored data stream or time series
Network-edge computing

Data Center, (Cloud)

Network Core

Multi-service Edge (Gateways, Edge routers)

Embedded Systems & Sensors (M2M devices)

Analyze and...

<table>
<thead>
<tr>
<th>T1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>T2</td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td></td>
</tr>
<tr>
<td>T4</td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td></td>
</tr>
<tr>
<td>T6</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Network-edge computing

Data Center, (Cloud)

Network Core

Multi-service Edge (Gateways, Edge routers)

Embedded Systems & Sensors (M2M devices)

Analyze and...
- Reduce

<table>
<thead>
<tr>
<th>T1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>T2</td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td></td>
</tr>
<tr>
<td>T4</td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td></td>
</tr>
<tr>
<td>T6</td>
<td></td>
</tr>
</tbody>
</table>

... ...
Network-edge computing

Data Center, (Cloud)

Network Core

Multi-service Edge (Gateways, Edge routers)

Embedded Systems & Sensors (M2M devices)

Analyze and...
- Reduce
- React

© NEC Corporation 2016
Data Center, (Cloud)

Network Core

Multi-service Edge (Gateways, Edge routers)

Embedded Systems & Sensors (M2M devices)

Network-edge computing

Analyze and...
- Reduce
- React
- Cache
Network-edge computing

Data Center, (Cloud)

Network Core

Multi-service Edge (Gateways, Edge routers)

Embedded Systems & Sensors (M2M devices)

Analyze and...
- Reduce
- React
- Cache
- ...

- T1
- T2
- T3
- T4
- T5
- T6
- ...
- T1
- T2
- T3
- T4
- T5
- T6
- ...

© NEC Corporation 2016
Why Network-edge computing?

Data Center, (Cloud)

1. Bandwidth
2. Network energy
3. I/O throughput
4. Data storage

Network Core

Multi-service Edge (Gateways, Edge routers)

Embedded Systems & Sensors (M2M devices)
NECtar (Edge Data Handling/Filtering solution)

Our solution for real-time per-item data reduction based on exchangeable data handlers and „streamified“ data reduction algorithms
Core Ideas

What do we do differently?

- “Streamification”
  - Developed **data reduction solutions that work upon data streams**, i.e., “per incoming item”, based on concepts of solutions that are currently designed to “compress” a posteriori, i.e., upon entire data sets

- Real-time aspect
  - **Reduced the “per item delay”** caused by the data handling at the edge by using cache reduction and cache projection techniques

- Reconstructability
  - Introduced “**reconstructability**” as data filtering criterion

- Exchangeable data handlers
  - Single-click **data handler instantiation** by implementing identical interfaces
NECtar Agent – Description of Operation

**Backend**

<table>
<thead>
<tr>
<th>Time</th>
<th>Data Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:00:00</td>
<td>17.9</td>
</tr>
<tr>
<td>12:00:10</td>
<td>18.1</td>
</tr>
<tr>
<td>12:00:20</td>
<td>20.2</td>
</tr>
</tbody>
</table>

**Network Edge Device**

<table>
<thead>
<tr>
<th>Handler</th>
<th>Type</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>h1</td>
<td>Sampling Handler</td>
<td>1:2 rates</td>
</tr>
<tr>
<td>h2</td>
<td>Sampling Handler</td>
<td>2:3 rates</td>
</tr>
<tr>
<td>h3</td>
<td>Important Points Handler</td>
<td>lows/highs</td>
</tr>
<tr>
<td>h4</td>
<td>Selective Forwarding Handler</td>
<td>values from list</td>
</tr>
<tr>
<td>h5</td>
<td>Selective Forwarding Handler</td>
<td>&gt;20</td>
</tr>
</tbody>
</table>

**Cache**

**Reconstructability Table**

<table>
<thead>
<tr>
<th>Handler</th>
<th>Reconstructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>h1</td>
<td>XX %</td>
</tr>
<tr>
<td>h2</td>
<td>YY %</td>
</tr>
<tr>
<td>h3</td>
<td>...</td>
</tr>
<tr>
<td>h4</td>
<td>...</td>
</tr>
<tr>
<td>h5</td>
<td>...</td>
</tr>
</tbody>
</table>

**GW application** (using the NECtar Agent)
Then we can apply and switch filtering logics as simply as...

```
BaseHandler h1
... 
```
```
h1 = new SamplingHandler(timeSeriesName, this, 2);
... 
```
```
h1.handleData();
... 
```

Classes that impl. the same interfaces fulfilling internally one of the data reduction algorithms
What is the problem?

- It is straightforward to apply sampling or approximation "per incoming item"...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm

- Simply explained:
Streamification

What is the problem?

- It is straightforward to apply sampling or approximation „per incoming item“...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm

- Simply explained:

```
value
```

Numbering the items in order of importance ...
Streamification

What is the problem?
- It is straightforward to apply sampling or approximation „per incoming item“...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm
- Simply explained:

Numbering the items in order of importance ...
What is the problem?
- It is straightforward to apply sampling or approximation “per incoming item”...
- BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm
- Simply explained:

Numbering the items in order of importance ...
Streamification

What is the problem?
- It is straightforward to apply sampling or approximation „per incoming item“...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm
- Simply explained:

Numbering the items in order of importance ...
Streamification

What is the problem?

- It is straightforward to apply sampling or approximation „per incoming item“...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm

- Simply explained:

![Diagram of Perceptually Important Points (PIP) algorithm](image)

*Numbering the items in order of importance ...*
What is the problem?

- It is straightforward to apply sampling or approximation „per incoming item“...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm

- Simply explained:

![Graph showing time and value with points numbered 1, 2, and 3.]

Numbers the items in order of importance...
Streamification

What is the problem?
- It is straightforward to apply sampling or approximation „per incoming item“...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm
- Simply explained:

![Graph showing Perceptually Important Points (PIP) algorithm]

Numbering the items in order of importance ...
Streamification

What is the problem?

- It is straightforward to apply sampling or approximation „per incoming item“...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm

- Simply explained:

![Diagram showing a graph with points numbered 1 to 4, with a trend line indicating value over time. The points are labeled with red circles, and a note box reads: "Numbering the items in order of importance ...". The text "Etc..." is placed in a red box in the diagram.](image-url)
Streamification

What is the problem?
- It is straightforward to apply sampling or approximation “per incoming item”...
- ...but it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm
- Simply explained:

So what happens when we try to apply this at the edge for an incoming item in real-time?
Streamification

What is the problem?
- It is straightforward to apply sampling or approximation „per incoming item“...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm
- Simply explained:

So what happens when we try to apply this at the edge for an incoming item in real-time?

Issues:
1. The data set is missing
Streamification

What is the problem?

- It is straightforward to apply sampling or approximation „per incoming item“...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm

- Simply explained:

So what happens when we try to apply this at the edge for an incoming item in real-time?

Issues:
1. The data set is missing
Streamification

What is the problem?

- It is straightforward to apply sampling or approximation „per incoming item”...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm

- Simply explained:

So what happens when we try to apply this at the edge for an incoming item in real-time?

Issues:
1. The data set is missing
2. Last item is always selected as most important
What is the problem?

- It is straightforward to apply sampling or approximation „per incoming item“...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm

- Simply explained:

---

So what happens when we try to apply this at the edge for an incoming item in real-time?

Issues:
1. The data set is missing
2. Last item is always selected as most important
Streamification

What is the problem?
- It is straightforward to apply sampling or approximation „per incoming item“...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm
- Simply explained:

So what happens when we try to apply this at the edge for an incoming item in real-time?

Issues:
1. The data set is missing
2. Last item is always selected as most important
Streamification

What is the problem?
- It is straightforward to apply sampling or approximation “per incoming item”...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm
- Simply explained:

So what happens when we try to apply this at the edge for an incoming item in real-time?

Issues:
1. The data set is missing
2. Last item is always selected as most important
3. How will the future look like?
Streamification

What is the problem?
- It is straightforward to apply sampling or approximation „per incoming item“…
- …BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm
- Simply explained:

So what happens when we try to apply this at the edge for an incoming item in real-time?

Issues:
1. The data set is missing
2. Last item is always selected as most important
3. How will the future look like?
4. How much time do I have for all this?
Streamification

What we did:

- A „real-time“ version of the PIP algorithm which
  - Uses a cache with a delay-aware time window as history
  - Uses cache projection into the future to add meaning to the measurement of important of the current item
  - Developed and evaluated three different cache projection strategies
    - CLONE: append a copy of the current item
    - TWIN: append a duplicate of the entire cache
    - AVG: append an item with an average value
  - Uses cache reduction to make the “per item processing delay” negligible compared to the transmission delay
  - Can be combined with a “requested reconstructability degree” in order to decide how important an item must be in order to be forwarded
  - (Please refer to our publications for details of the algorithms...)
Network-edge data filtering evaluation summary

1. Complex analytics performed “per-item” and in adjustable time (down to <30 ms)

2. Reconstructabilities >90% through appropriate handler activation by forwarding only 20-30% of the items

3. Libraries of flexible, adjustable data handlers

(Cloud) Backend

Instantiated data handlers

Network Edge Device

Libary of handlers
- SamplingHandler
- PIPHandler

Data Source

Cache

Reconstructability Table
- Handler: h1, Reconstr: XX %
- Handler: h2, Reconstr: YY %
- Handler: h3, Reconstr: ...
Edge deployment of IoT data streaming tasks
Stream Processing Frameworks functionality

Developers provide...

- Computation Topology descriptions
- Deployable Implementations of Components
- Network Topology descriptions
- Deployment Settings, Preferences, Restrictions

...for...

Stream Processing Framework

- SPF monitor (network links, nodes, topology traffic)
- SPF extensions (analyzers, schedulers, deployment optimizers)

...deployment on processing nodes
Gap analysis

- SPFs are designed for performing stream processing in the Cloud

- In terms of task allocation and execution, standard SPFs ignore:
  - node heterogeneity
  - geo-distributed nature of IoT data sources
  - special data traffic and delay requirements
  - criticality of certain sensors and actuators

- In many cases edge computing can help, BUT this is not indicated by parameters that stream processing frameworks usually see

- For example...
Example surveillance topology with topology-external interactions

**NOTE:** Tasks can be instantiated as many times as required and their instances can be deployed on any of the Edge or Cloud nodes.

<table>
<thead>
<tr>
<th>Space</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>t1</strong></td>
<td>Camera resolution increase Latency</td>
</tr>
<tr>
<td></td>
<td>Time required from the moment Task2 has received a frame with many (unclear) faces until the moment that Task2 has issued the „resolution increase command“ to the IP camera</td>
</tr>
<tr>
<td><strong>b1</strong></td>
<td>Cloud-edge bandwidth consumption</td>
</tr>
<tr>
<td></td>
<td>Amount of data traversing the Cloud-edge NW (per second), e.g., the sum of Task2-&gt;BackendDB and the Task2-Task3 traffic if Task1 and Task2 run on edge nodes and Task3 runs on Cloud nodes (or the sum of Task1-&gt;Task2 and Task2-&gt;OnSiteDB traffic, if Task2 is moved to the Cloud etc.)</td>
</tr>
</tbody>
</table>
There are **three main things** (categories of characteristics) that shall **determine if a task is relevant to network edge computing** (and shall be executed at the edge) or not. These are:

- The **interfaces of the task with the environment**, i.e., control of actuators, direct provision of intermediate results to users, event- or alarm-raising.

- The characteristics of the **databases** with which the task interacts.

- The **task computation characteristics**, namely its CPU- and data-intensity and security restrictions.

```json
{  "edgeInteractions": {    "edgeExecutionRequired": "no",    "controlledActuators": [      {        "name": "actuator1",        "type": "switch",        "area": "area1",        "geo": "coordinates",        "latencyRequirement": "low/medium/high"      },      {        "name": "actuator2",        "type": "switch",        "area": "area2",        "geo": "coordinates",        "latencyRequirement": "low/medium/high"      }    ],    "provisionedIntermediateResults": [      {        "result": "res1",        "access": "pubSub",        "mainUsersArea": "area1",        "mainUsersLayer": "edge/core/cloud",        "latencyRequirement": "low/medium/high"      },      {        "result": "res2",        "access": "pubSub/pub2",        "mainUsersArea": "area2",        "mainUsersLayer": "edge/core/cloud",        "latencyRequirement": "low/medium/high"      }    ],    "potentiallyRaisedEvents": [      {        "event": "event1",        "propagationArea": "area1",        "propagationLayer": "edge/core/cloud",        "latencyRequirement": "low/medium/high"      },      {        "event": "event2",        "propagationArea": "area2",        "propagationLayer": "edge/core/cloud",        "latencyRequirement": "low/medium/high"      }    ]  },  "editedDatabases": [    {      "name": "db1",      "type": "rdbs",      "area": "area1",      "dbLayer": "edge/core/cloud",      "mainUsersLayer": "edge/core/cloud",      "latencyRequirement": "low/medium/high"    },    {      "name": "db2",      "type": "rdbs/mysql",      "area": "area2",      "dbLayer": "edge/core/cloud",      "mainUsersLayer": "edge/core/cloud",      "latencyRequirement": "low/medium/high"    }  ],  "computationCharacteristics": {    "cpuIntensity": "low/medium/high",    "expectedDataIntIntesity": "low/medium/high",    "forbiddenAreas": ["area1", "area2", "area3"],    "forbiddenLayers": ["layer1", "layer2", "layer3"],    "forbiddenDomains": ["domain1", "domain2", "domain3"],    "specificDevice": "targetDeviceName"  }  }
```
Implementation and evaluation summary

We implemented our „edge-aware SPF” concept as an extension of Apache Storm, evaluated it against Storm, and tested it with example topologies...

Latency violations:

Used Cloud-Edge bandwidth:
Conclusion
Conclusion

Data Filtering

Cloud
(servers, data centers)

Network Core
(routers)

Multi-service Edge
(gateways, edge routers, mini servers, mini data centers)

IoT devices
(embedded systems, sensors, personal devices)

State-of-the-art challenge: The most efficient time series reduction algorithms can work only a posteriori upon complete data sets. **We:** „Streamify“ such algorithms.

State-of-the-art challenge: Information loss might often demotivate data reduction. **We:** Consider data „reconstructability“ and Importance before reducing.

State-of-the-art challenge: Dynamic switching between different types of network edge processing can be heavy or complex for gateway middleware. **We:** Develop flexible, adjustable data handlers enabling fast system adaptation.

Edge-aware task deployment

- Consider edge computing characteristics such as...
  - Critical actuations, DB interactions, user locations, IoT node characteristics, system usage
- ...in order to place tasks of IoT processing chains at the right “edges”
NEC brings together and integrates technology and expertise to create the ICT-enabled society of tomorrow.

We collaborate closely with partners and customers around the world, orchestrating each project to ensure all its parts are fine-tuned to local needs.

Every day, our innovative solutions for society contribute to greater safety, security, efficiency and equality, and enable people to live brighter lives.