TECHNISCHE UNIVERSITÄT BERLIN

Data Systems Project

Kick-off Presentation

Linux Process Source & Sink

Connecting NebulaStream to the Linux Ecosystem

Team Members:

Saad Aldeen Mnowar	495171
Roshni Ajay Melwani	475317
Emi Maliqi	498273
Aleksandar Georgiev	468867

1. Challenge and Vision

Current State

NebulaStream primarily ingests data from static files or network sockets.

Therefore, there are limitations to direct interaction with local system tools.

NebulaStream

Objective

Enable NebulaStream to pipe data directly to and from standard Linux processes (scripts) via Standard Streams.

2. System Architecture Overview

Input Side

Execute external scripts and ingest their Standard Output (stdout) directly into the streaming engine.

Processing

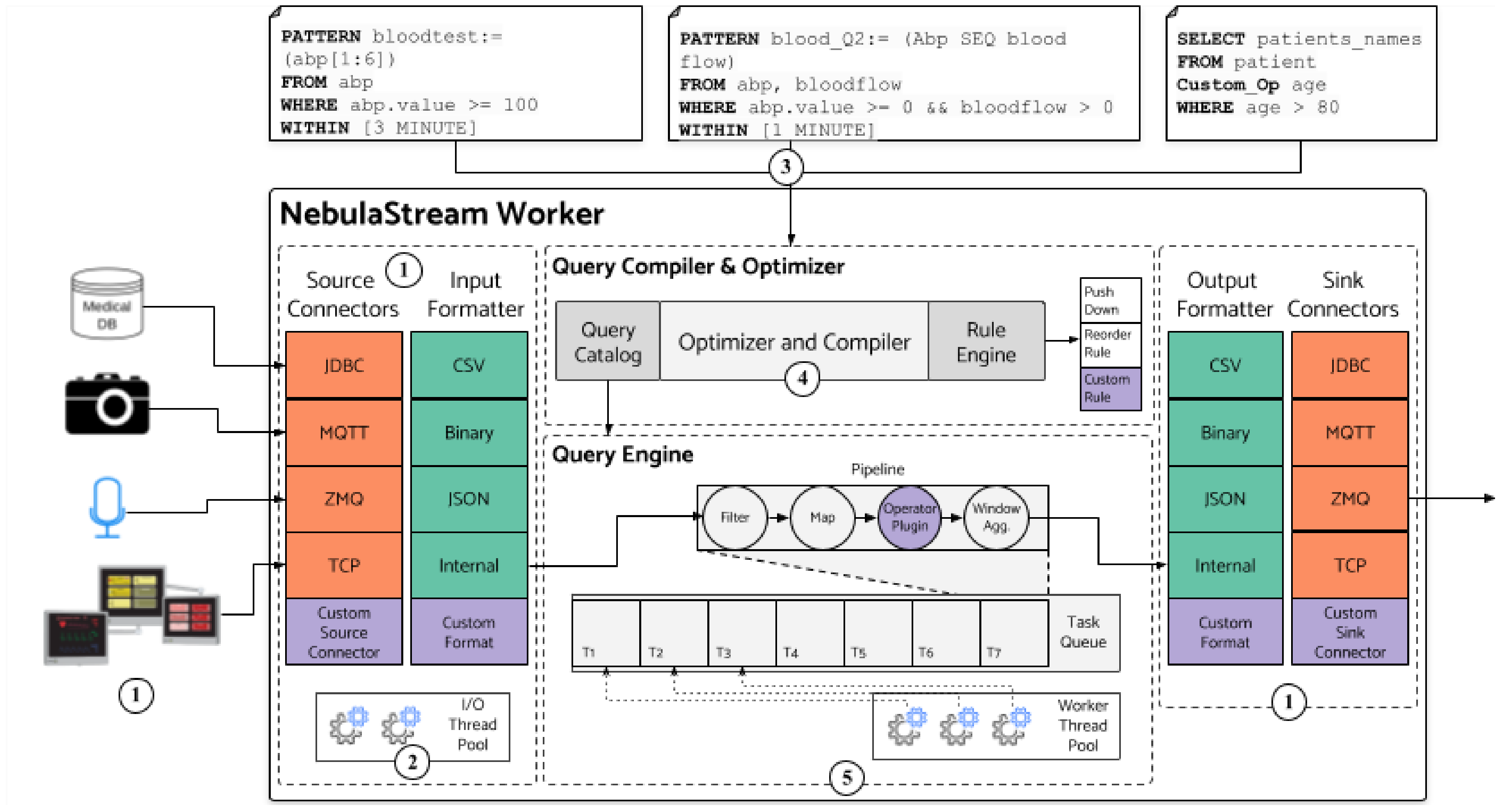
NebulaStream performs real-time analytics and transformations on the incoming data stream.

Output Side

Pipe processed results into the Standard Input (stdin) of downstream Linux tools for further use.

2. System Architecture Overview

Source: https://nebula.stream/


```
PATTERN bloodtest:=
(abp[1:6])
FROM abp
WHERE abp.value >= 100
WITHIN [3 MINUTE]
```

```
PATTERN blood_Q2:= (Abp SEQ blood
flow)
FROM abp, bloodflow
WHERE abp.value >= 0 && bloodflow > 0
WITHIN [1 MINUTE]
```

```
SELECT patients_names
FROM patient
Custom_Op age
WHERE age > 80
```

3. Implementation Plan: Source (Input) New Plugin

New Plugin Creation

Create LinuxProcessSource within the nes-plugins module as the foundation for process input handling.

Configuration Layer

Implement a robust configuration descriptor to accept arbitrary command strings (e.g., "python3 sensor.py").

Process Mechanism

Utilize popen() to spawn child processes and establish a read-only pipe for data capture.

Data Ingestion Logic

Implement fillTupleBuffer to continuously read stdout bytes and parse them into NebulaStream Tuples.

4. Implementation Plan: Sink (Output) New Plugin

New Plugin Creation

Create LinuxProcessSink mirroring the source structure.

Process Mechanism

Utilize popen() to establish a write-only pipe to the target process.

Data Formatting

Leverage existing formatters (CSV/JSON) to convert internal binary Tuples back into text streams.

Data Ingestion Logic

Implement consumeTupleBuffer to write formatted data directly to the child process's stdin.

5. Testing & Validation Strategy

Unit Tests

- Test individual components in isolation
- Validate buffer operations
- Verify data parsing accuracy
- Check process lifecycle management

Error Handling

- Process termination scenarios
- Pipe failure recovery
- Malformed data handling
- Resource cleanup verification

Integration Tests

- End-to-end pipeline validation
- Performance benchmarking
- Compatibility with common tools
- Stress testing under load

TECHNISCHE
UNIVERSITÄT
BERLIN

Thank you for the attention!