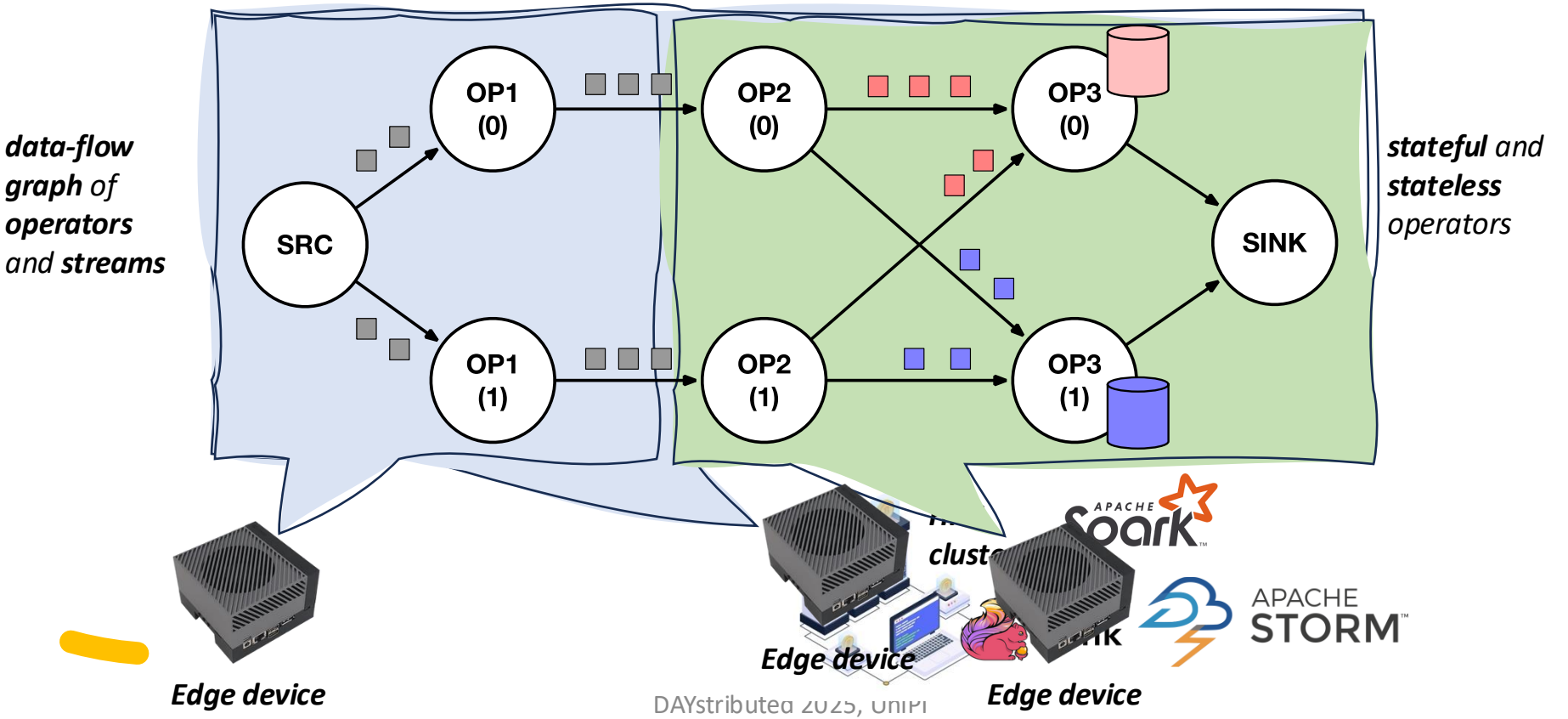
Efficient Stream Processing on Resource-Constrained Devices

Contributors: G. Mencagli, M. Danelutto, P. Dazzi, and M. Torquati
10+ students involved over the years (Bachelor, Master)

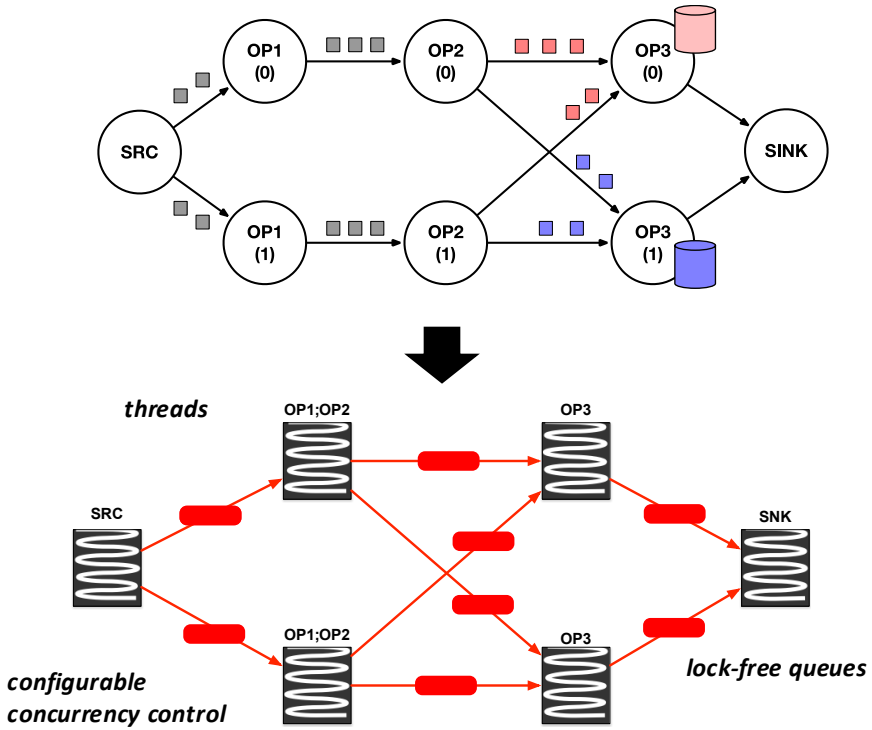
Stream Processing + Edge Resources

Computing paradigm characterized by the continuous analysis of data streams (e.g., finding insights, knowledge, analytics)



WindFlow Library

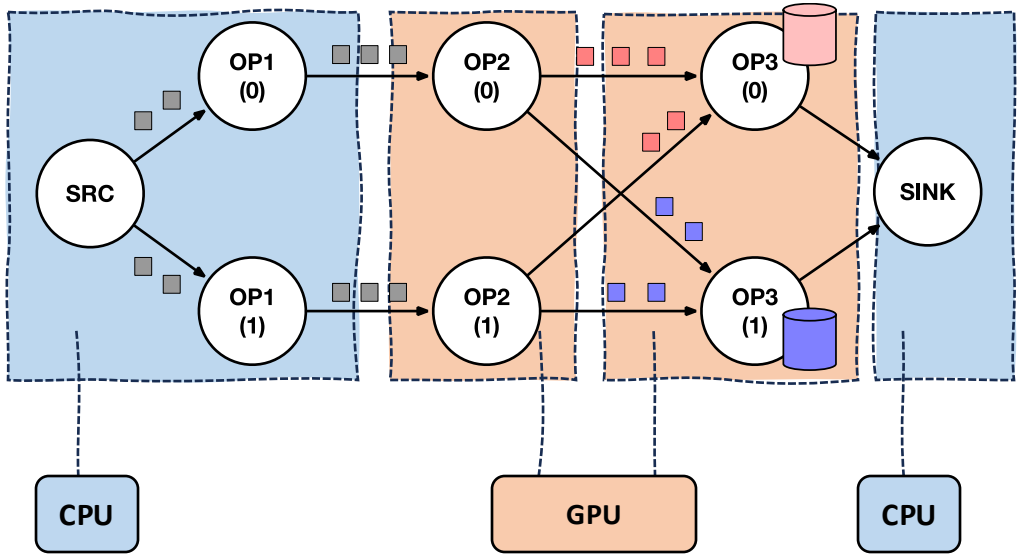
C++17 library written on top of the **FastFlow** parallel programming framework

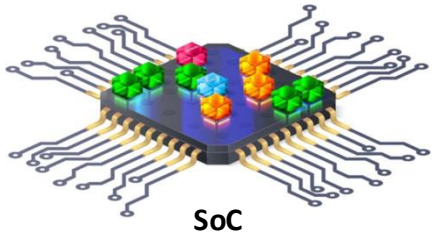



```
PipeGraph app;
Source src = Source_Builder(…)
                 .withParallelism(2)
                 .build();

Map op1 = Map_Builder([](input_t &t) -> void {…})
                 .withParallelism(2)
                 .build();

FlatMap op2 = FlatMap_Builder([](input_t &t, Shipper<output_t> &s) {…}
                 .withParallelism(2)
                 .build();

Aggregation op3 = Aggregation_Builder([](const output_t &t1,
                                         const output_t &t2) -> output_t {…})
                 .withParallelism(2)
                 .withKeyBy([](const output_t &t) -> key_t {…})
                 .withTimeBoundaries(seconds(10), seconds(1))
                 .build();

Sink sink = Sink_Builder(…)
                 .withParallelism(2)
                 .build();

app.add_source(src).add(op1).add(op2).add(op3).add_sink(snk);
```

Hardware Accelerators

Edge resources as complex **System-on-Chip** devices (CPU+GPU+FPGA+ ...)

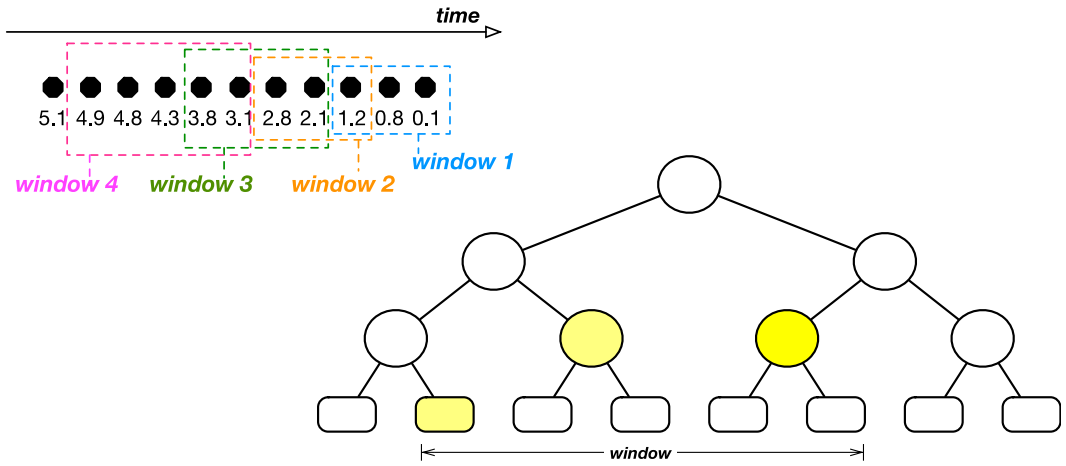




GPU-accelerated operators (both **stateless** and **stateful**)

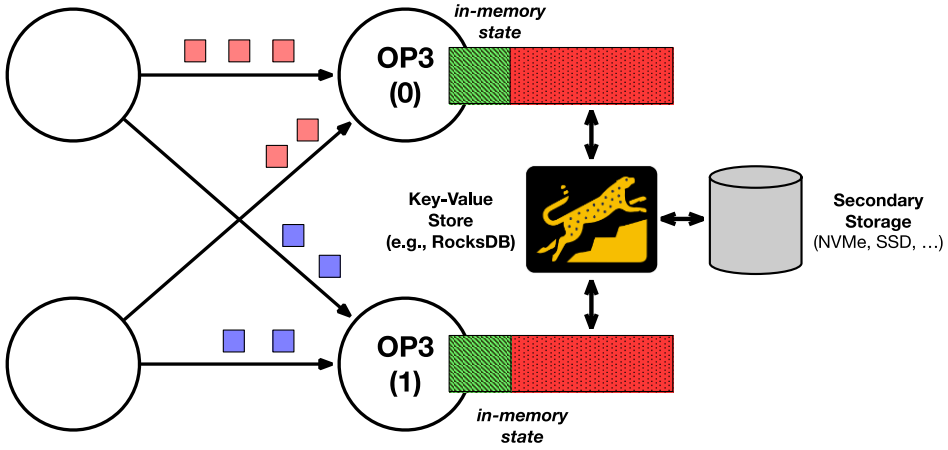
- **Micro-batching**
- **Kernels** defined in the runtime system, user provides **device functions** called by the kernels
- Automatic **H2D** and **D2H data transfers** performed with high overlapping capabilities

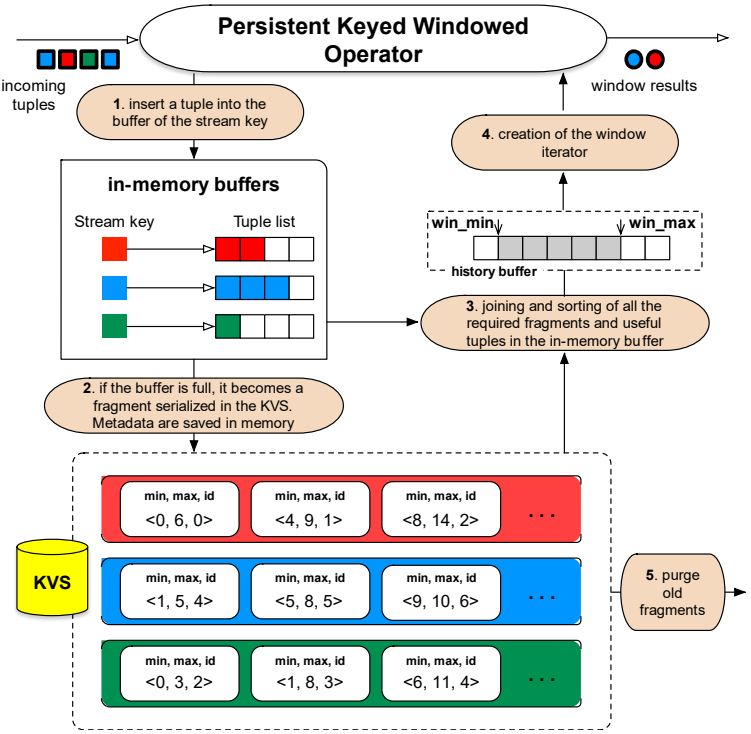
Stateful operators are challenging on GPUs



Larger-than-Memory Stateful Processing

- Large aggregations over temporal windows
- State might exceed the available memory (especially on Edge resources)





Different layouts to represent the state on a KVS

- **Fragment-based** *vs* **window-centric** layouts
- **Keyed** *vs* **unkeyed** state objects

Trade-offs between **performance**-**memory footprint**-**secondary memory usage**

Research Ideas for the Future

- **Dynamic** selection of **CPU-based** or **GPU-based operators** based on the workload conditions
- Adapt the **machine configuration** (e.g., DVFS, idle cores, GPU freq) based on the actual workload
- Use of **lossless compression** to further reduce the state size
- Use of WindFlow as a highly efficient system for **ML inference**
  - **Why?** WindFlow provides advanced functionalities for online stream transformations, aggregation, sampling, windowing but lacks support for **ML model management**. However, ML tools such as **TensorFlow**, **PyTorch** provide limited support to streaming
  - **How?** Embed ML models into WindFlow (e.g., interoperability with **external libraries** such as **ONNX Runtime**), or alternatively use external **serving frameworks** such as **TensorFlow Serving** or **TorchServe**