Advanced visualization of Flink and Spark jobs

Zoltán Zvara
zoltan.zvara@sztaki.hu

Márton Balassi
mbalassi@apache.org

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 688191.

Agenda

- Introduction
- Motivation and the challenges
- Preliminaries to execution visualization
- An elegant way to tackle data skew
- The visualizer extended to Flink and Spark
- Future plans

MTA SZTAKI

Introduction

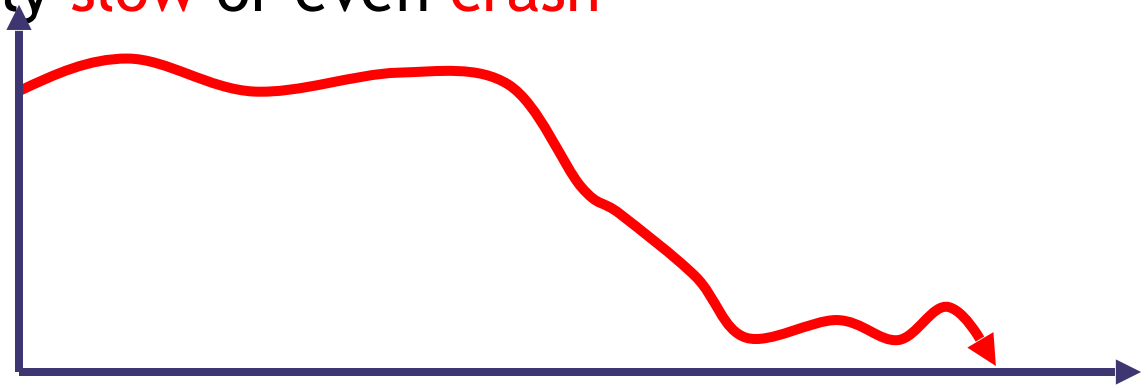
- Hungarian Academy of Sciences, Institute for Computer Science and Control (MTA SZTAKI)
- Research institute with strong industry ties
- Big Data projects using Flink, Couchbase, Spark, Hadoop YARN etc.

STREAMLINE.

- Multiple telco use cases lately, with challenging data volume and **distribution**

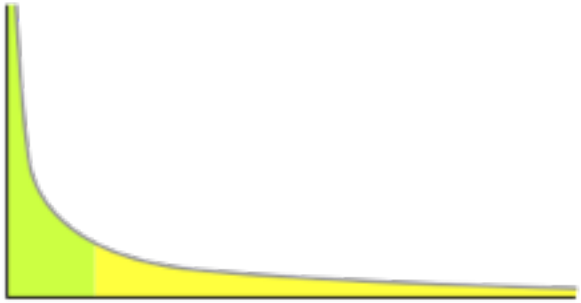
Motivation

- We have worked on many telco use cases lately, with challenging data volume and distribution
- We have developed an application aggregating telco data that tested well on toy data
- When deploying it against the real dataset the application seemed healthy
- However it could become surprisingly slow or even crash
- What did go wrong?



Data skew

- When some data-points are very frequent, and we need to process them on a single machine – aka the *Phelps-effect*
- Power-law distributions are very common: from social-media to telecommunication, even in our beloved WordCount
- Default hashing puts these into „random” buckets
- We can do better than that

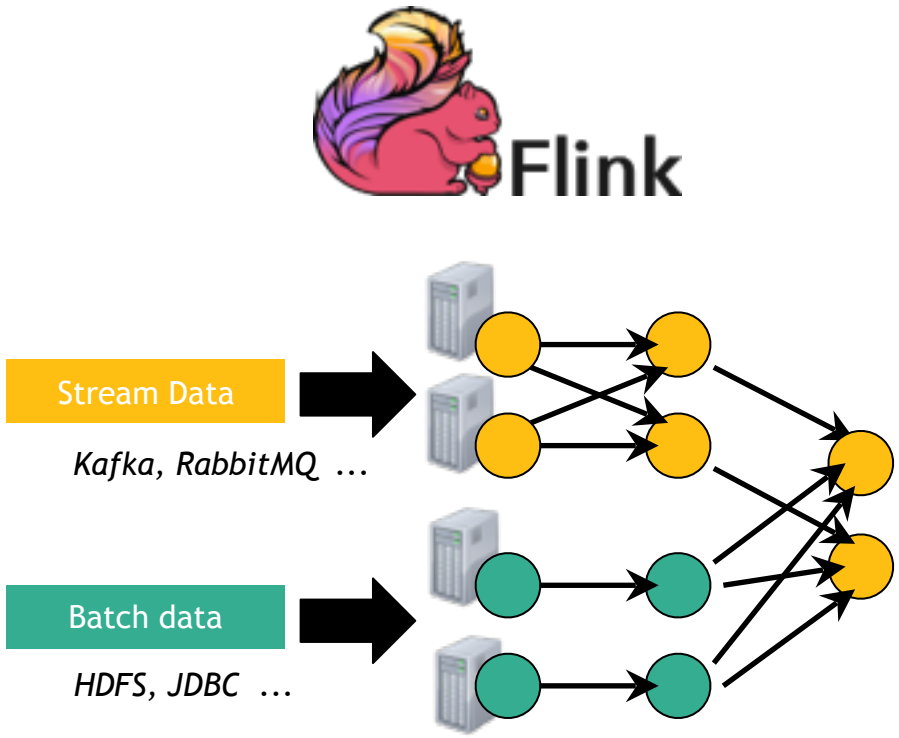


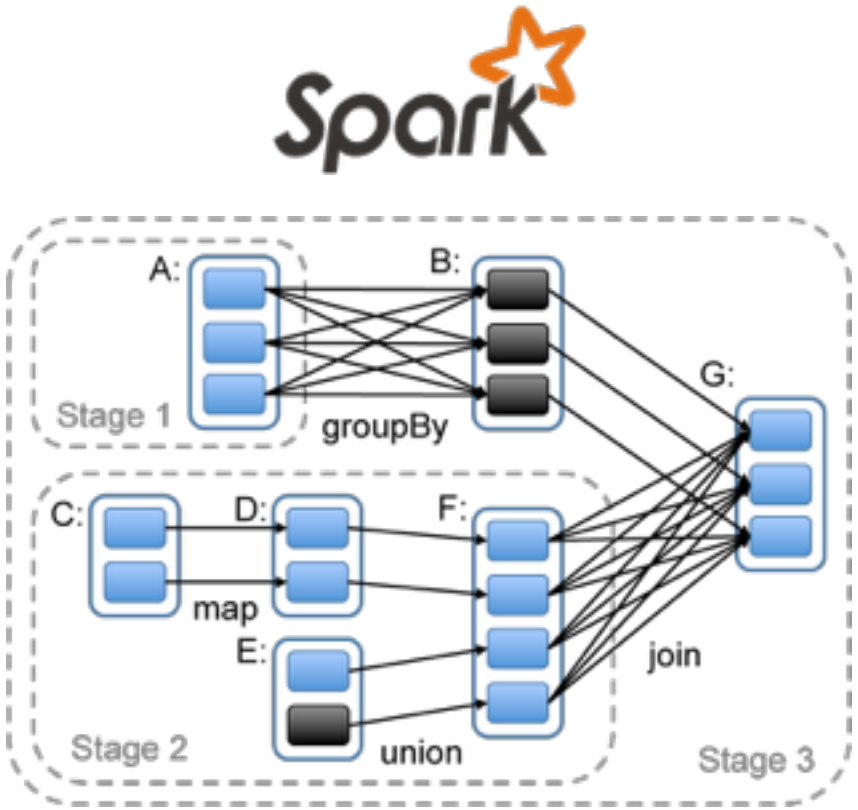
MTA SZTAKI

Aim

1. Most of the telco data-processing workloads suffer from inefficient communication patterns, introduced by skewed data
2. Help developers to write better applications by detecting issues of logical and physical execution plans easier
3. Help newcomers to understand a distributed data-processing system faster
4. Demonstrate and guide the testing of adaptive (re)partitioning techniques

Computational models



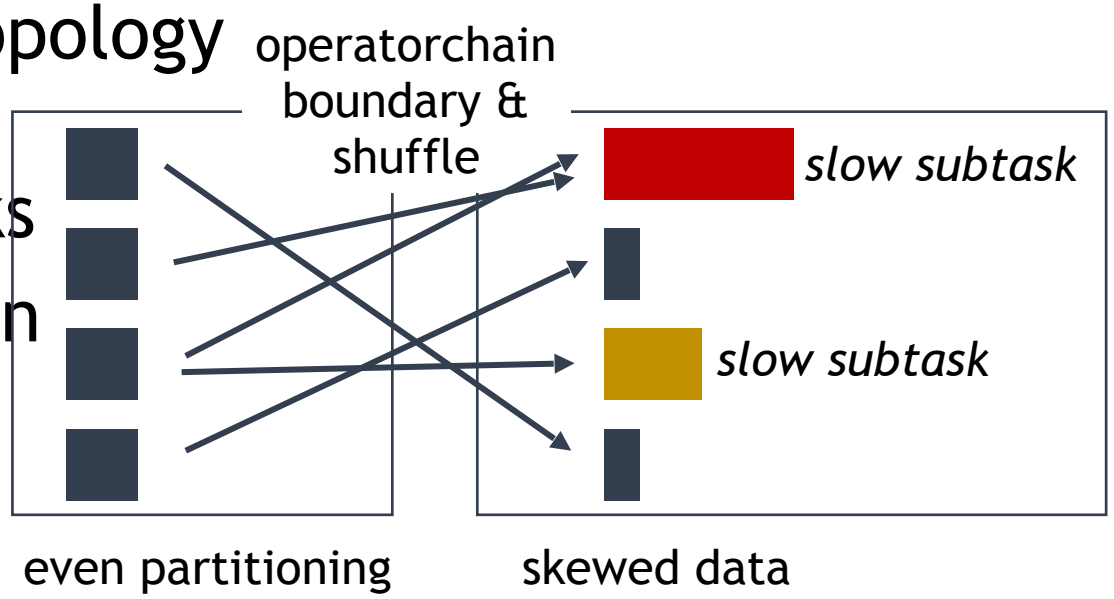


Flink computation is fully pipelined by default

Spark RDDs break down the computation into stages

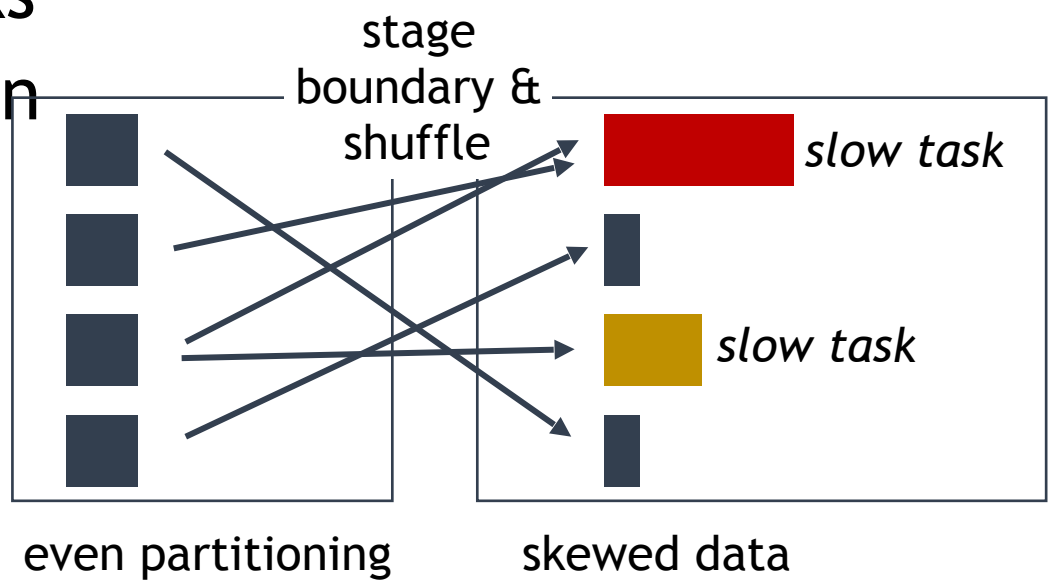
Flink's execution model

- Distributed dataflow model
- Batch execution is pipelined by default, can be broken up into stages; Tasks get scheduled on first input
- In streaming all the tasks in the topology get scheduled on submit
- Data skew can introduce slow tasks
- In most cases, the data distribution is not known in advance
- „Concept drifts" are common



Spark's execution model

- Extended MapReduce model, where a previous stage has to complete all its tasks before the next stage can be scheduled (global synchronization)
- Data skew can introduce slow tasks
- In most cases, the data distribution is not known in advance
- „Concept drifts" are common

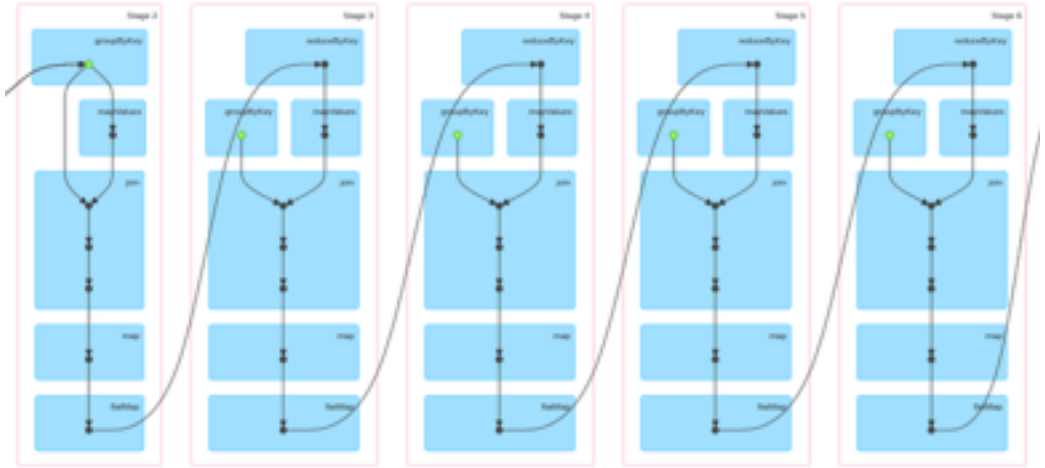


Physical execution plan

- Understanding the physical plan
  - can lead to designing a better application
  - can uncover bottlenecks that are hard to identify otherwise
  - can help to pick the right tool for the job at hand
  - can be quite difficult for newcomers
  - can give insights to new, adaptive, on-the-fly partitioning & optimization strategies

Current visualizations in Spark & Flink

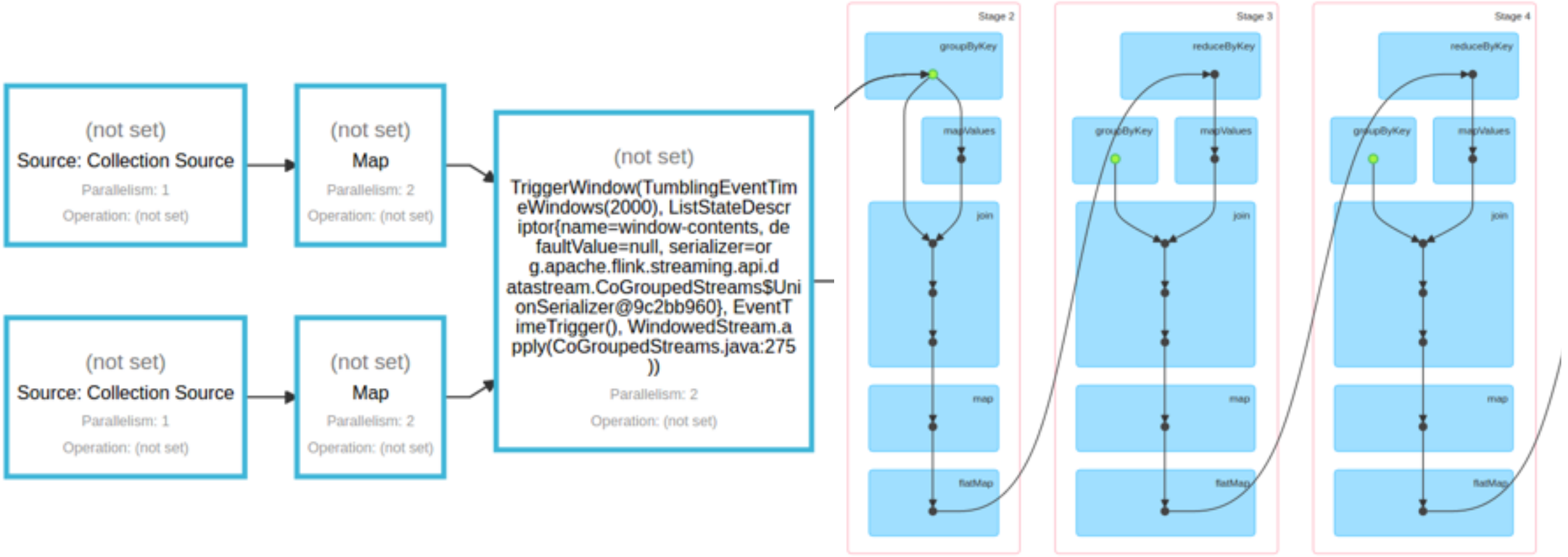
- Current visualization tools
  - are missing the fine-grained input & output metrics – also not available in most sytems, like Flink
  - does not capture the data charateristics – hard to accomplish a lightweight and efficient solution
  - does not visualize the physical plan



DAG visualization in Spark

MTA SZTAKI

The currently available visualizers



Flink

Spark

MTA SZTAKI

New metrics in Flink

- Current metric collection is tied to Serialization stack
  - It collects read and write metrics
- We have made the metric collection more fine grained
  - Instead of collecting on the operator level
    we need information on the inputchannel level
- We distinguish pointwise and all-to-all and distribution patterns

```
"granular-metrics":{
     "read bytes":{"0":12,"1":178,"2":24,"3":198},
     ...
}
```

New metrics in Spark

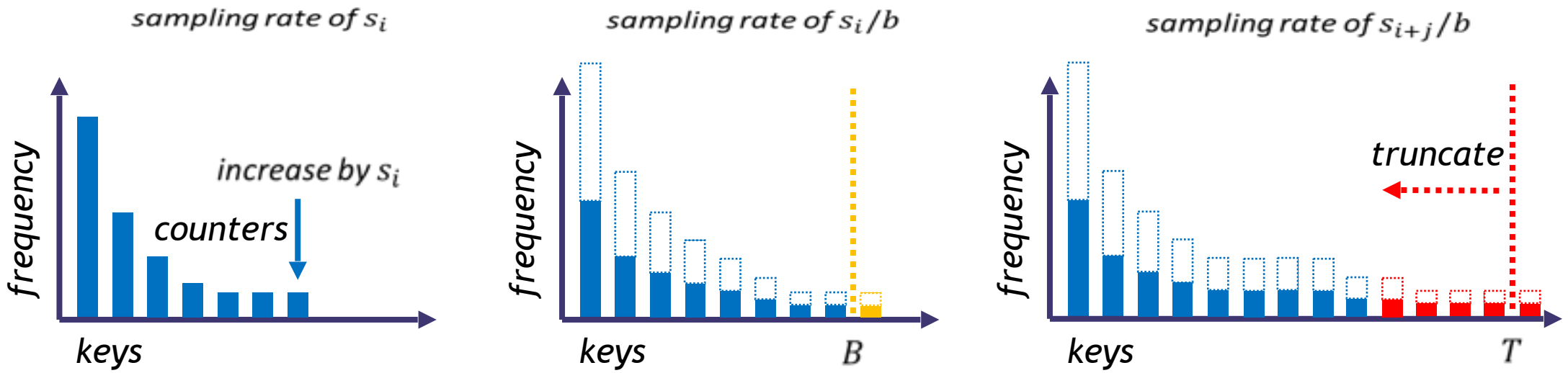
- Data-transfer between tasks are accomplished in the shuffle phase, in the form of shuffle block fet
  - We distinguish local & remote block fetches
- Data characteristics collected on shuffle write &

```
recordsRead: 8399,
dataCharacteristics: {
    3424: 19.75,
    115752: 32.25,
    204710: 19.75,
    254186: 17.25
}
```

```
remoteBlocksFetched: 0,
remoteBlockFetchInfos: [ ],
localBlocksFetched: 10,
localBlockFetchInfos: [
  - {
      - blockId: {
            shuffleId: 6,
            mapId: 0,
            reduceId: 7,
            shuffle: true,
            rdd: false,
            broadcast: false
        },
        bytes: 871162
    },
  - {
      - blockId: {
            shuffleId: 6,
            mapId: 1,
            reduceId: 7,
            shuffle: true,
            rdd: false,
            broadcast: false
        },
        bytes: 872696
    },
```

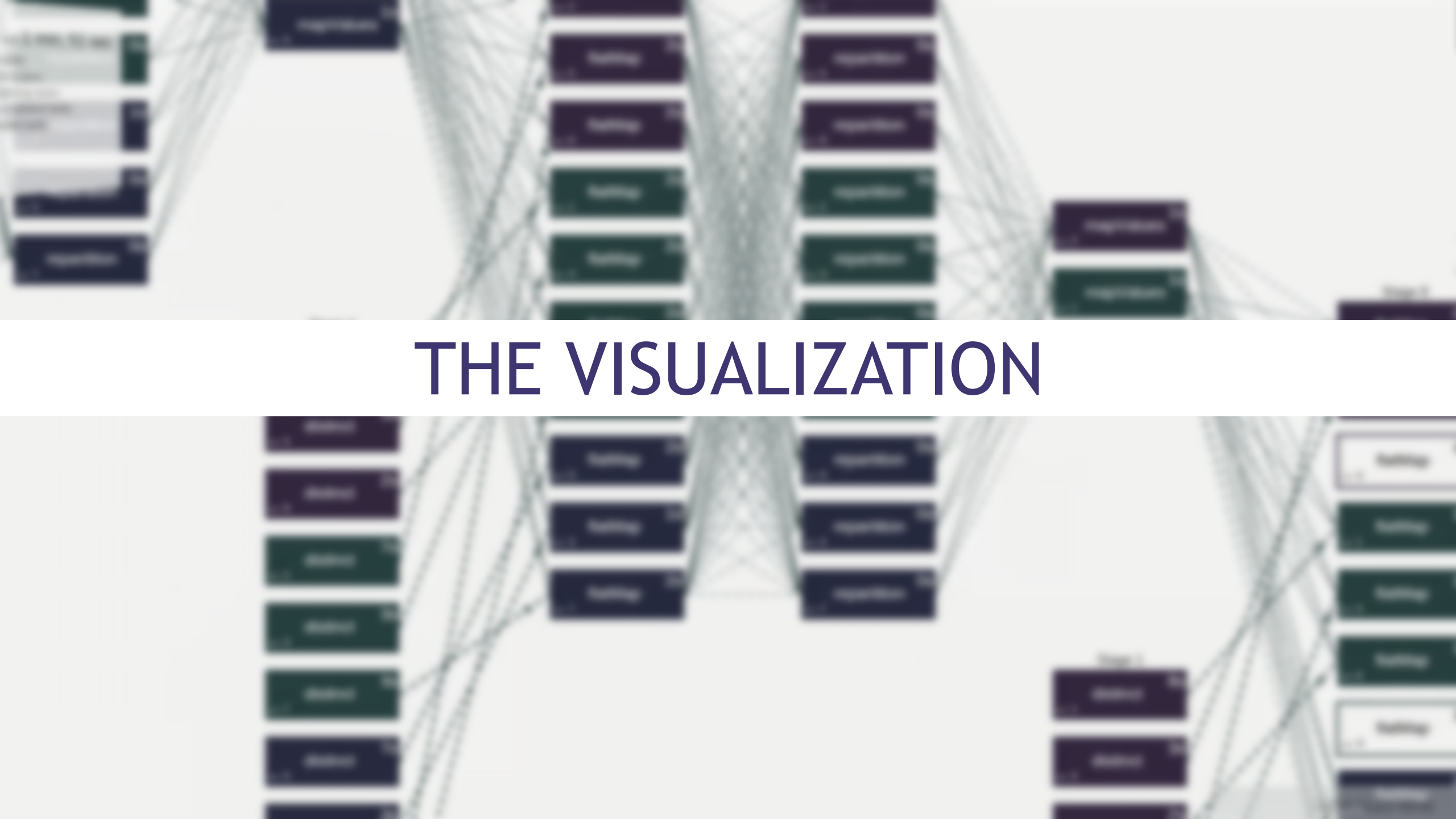
A scalable sampling

- Key-distributions are approximated with a strategy, that
  - is not sensitive to early or late concept drifts,
  - lightweight and efficient,
  - scalable by using a backoff strategy



Availability through the Spark & Flink REST API

- Enhanced REST APIs to provide block fetch & data characteristics information of tasks (Spark), communication that occurs between sub-tasks of vertices (Flink)
- New queries to the REST API, for example: „what happend in the last 3 seconds?”
- `/api/v1/applications/{appId}/{jobId}/tasks/{timestamp}`



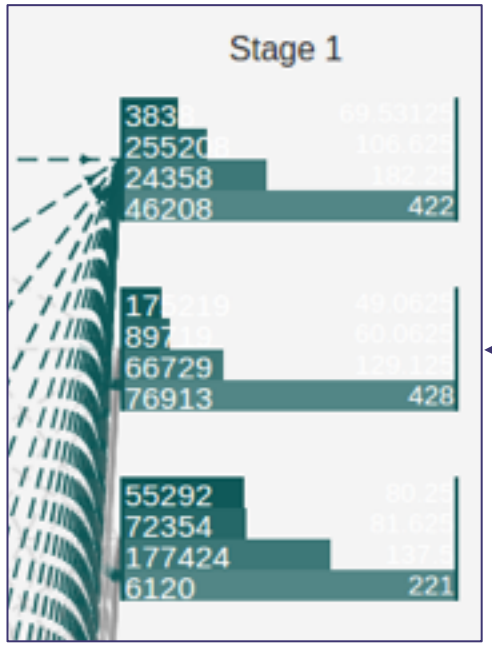
THE VISUALIZATION

ME 29 sec
des
ecutors
nning tasks
mpleted tasks
led tasks

DROP FILES HERE

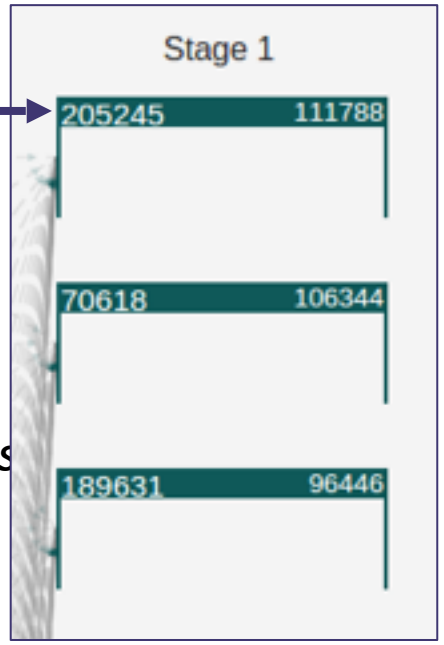
Our way of tackling data skew

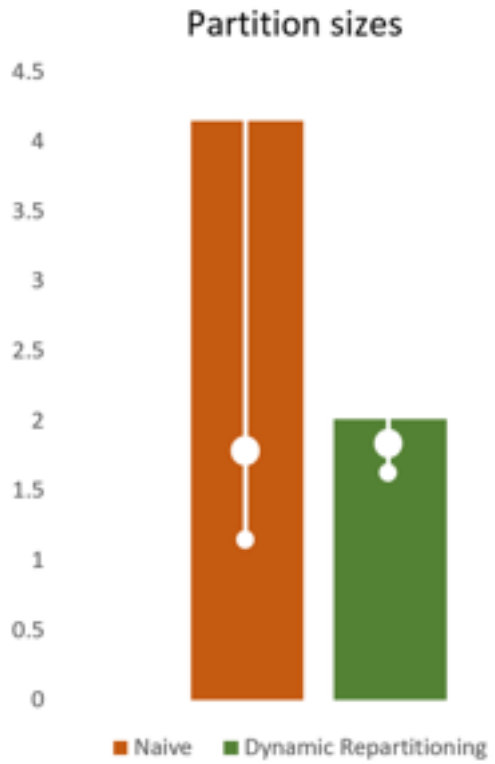
- Goal: use the collected information to handle data skew dynamically, on-the-fly on any workload (batch or streaming)



Phelps-key is alone, size of the biggest partition is minimized

heavy keys create heavy partitions (slow tasks





MTA SZTAKI

Future plans

- Visual improvements, more features
- Public beta is going to be available in the near future
- Adapt Dynamic Repartitioning & Data Awareness to other systems as well
- Opening PRs against Spark & Flink with the suggested metrics

MTA SZTAKI

Conclusions

- The devil is in the details
- Visualizations can aid developers to better understand issues and bottlenecks of certain workloads
- Data skew can hinder the completion time of the whole stage in a batch engine
- Adaptive repartitioning strategies can be suprisingly efficient

STREAMLINE.

Thank you for your attention

Q&A

Zoltán Zvara
zoltan.zvara@sztaki.hu

Márton Balassi
mbalassi@apache.org