Data Stream Management

Big Data Management

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH
DTIM www.essi.upc.edu/dtim

Knowledge objectives

1. Define a data stream
2. Distinguish the two kinds of stream management systems
3. Recognize the relevance of stream management
4. Enumerate the most relevant characteristics of streams
5. Explain to which extent a DBMS can manage streams
6. Name 10 differences between DBMS and SPS
7. Characterize the kinds of queries in an SPS
8. Explain the two parameters of a sliding window
9. Explain the three architectural patterns
10. Explain the goals of Spark streaming architecture
11. Draw the architecture of Spark streaming

Understanding Objectives

1. Identify the need of a stream ingestion pattern
2. Identify the need of a near-real time processing pattern
3. Identify the kind of message exchange pattern
4. Simulate the mesh-join algorithm
5. Estimate the cost of the mesh-join algorithm
6. Use windowing transformations in Spark streaming

Basics

Tens of thousands of elements/events per second

- Internet traffic analysis
- Trading on Wall Street
- Fraud detection (i.e., credit cards)
- Highway traffic monitoring
- Surveillance cameras
- Command and control in military environments
- Log monitoring
  - Google receives several hundred million search queries per day
- Click analysis
  - Yahoo! Accepts billions of clicks per day
- Scientific data processing (i.e., sensor data)
  - One million sensors reporting at a rate of ten per second would generate 3.5TB/day (only 4 bytes per message)
- RFID monitoring
  - Venture Development Corporation predicted in 2006 that RFID can generate in Walmart up to 7TB/day (≈ 292GB/hour ≈ 5GB/minute ≈ 80MB/second)

Stream characterization

1. Arrival rate not under the control of the system
   - Faster than processing time -- algorithms must work with only one pass of the data
2. Unbounded memory requirements -- Some drastic reduction is needed
3. Keep the data moving -- Only volatile storage
4. Support for near-real time application -- Latency of 1 second is unacceptable
   - Need to scale and parallelize
5. Arrival order not guaranteed -- Some data may be delayed
6. Imperfections must be assumed -- Some data will be missing
7. There is temporal locality -- Data (characteristics) evolve over time
8. **Approximate** (not accurate) **answers** are acceptable
   - Deterministic outputs

Data Stream Management System (DSMS)

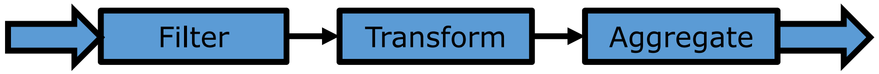
"Class of software systems that deals with processing streams of high volume messages with very low latency"

M. Stonebraker

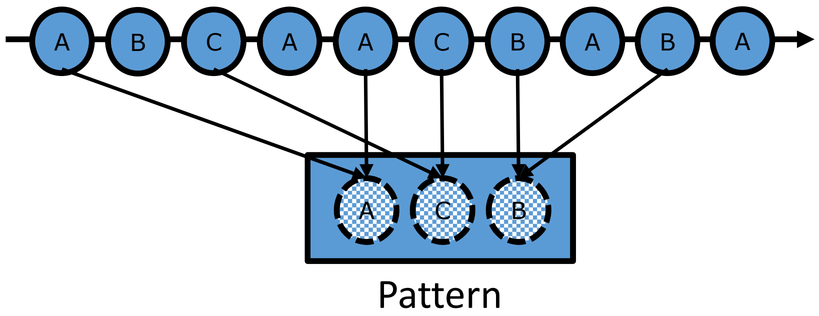
- Messages constantly arrive at a high pace
- Sub-second latency

Kinds of systems

- Stream Processing Engine
  - Focus on
    - Near-real time processing and scalability
    - Offering windowing operations to define aggregates
  - Tools
    - Spark streaming
    - Flink
    - Storm
    - S4



- Complex Event Processing/Pattern Matching Engine
  - Focus on
    - Offering windowing operations to define indicators (based on thresholds)
    - Express complex temporal correlations
  - Tools
    - Esper
    - Aleri
    - StreamBase
    - T-Rex
    - Huawey PME
    - Orange CRS network monitoring



SPE vs CEP

Stream Processing Engine	Complex Event Processing
Keep data moving	**Pattern** identification
Window aggregates definition	**Pattern** expressions
Handle stream imperfections	
Integrate stored and streamed data	**State** management
High availability of **processing**	High availability of **patterns**
Process distribution	**States** distribution

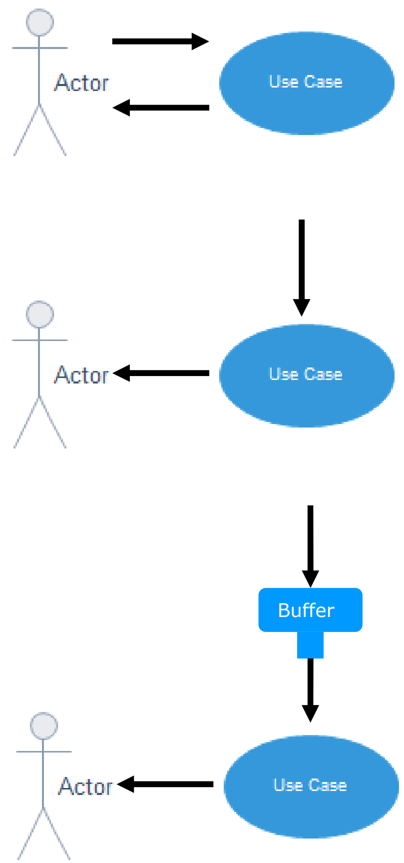
Characterization of operations in SPE

Kinds of system interaction

CRUD

Stream

Micro-batch



Kinds of queries

- Depending on the trigger
  - Standing
  - Ad-hoc
- Depending on the output
  - Alerts
  - Result set
- Depending on the inputs
  - Based on the X last elements
    - Based on the last element
    - Sliding window
  - Based on a summary
    - Synopsis/Sketches

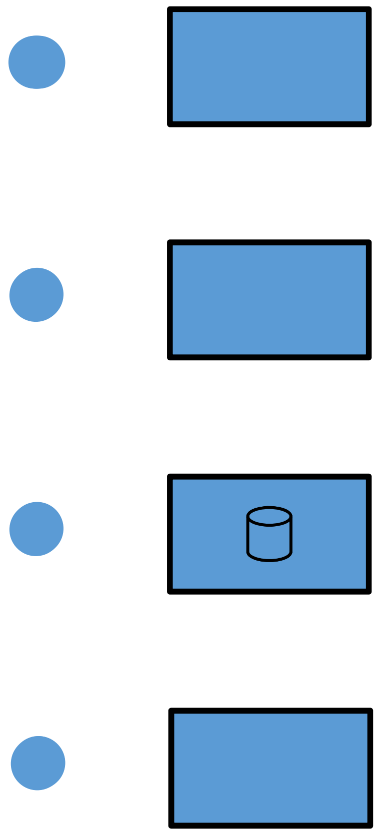
Kinds of operations

Filter

Project

Lookup

Aggregation



Tumbling&Sliding window examples

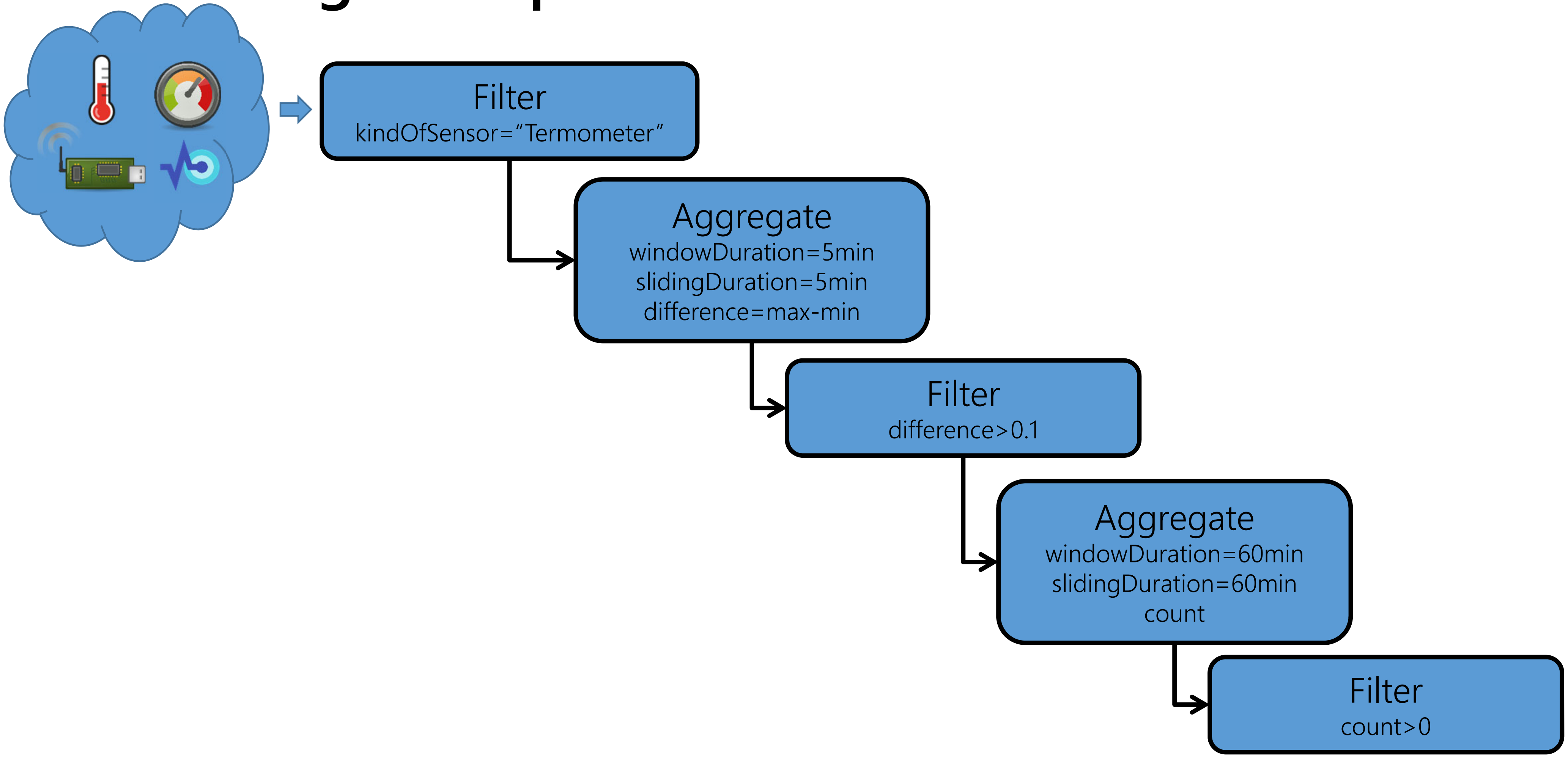
WINDOW DURATION = 5
SLIDING DURATION = 5

20.6 20.5 20.6 20.5 20.5	20.5 20.4 20.4 20.3 20.2	20.1 20.0 20.1 20.1 20.2	20.1 20.0 19.9 20.0 20.1

WINDOW DURATION = 5
SLIDING DURATION = 3

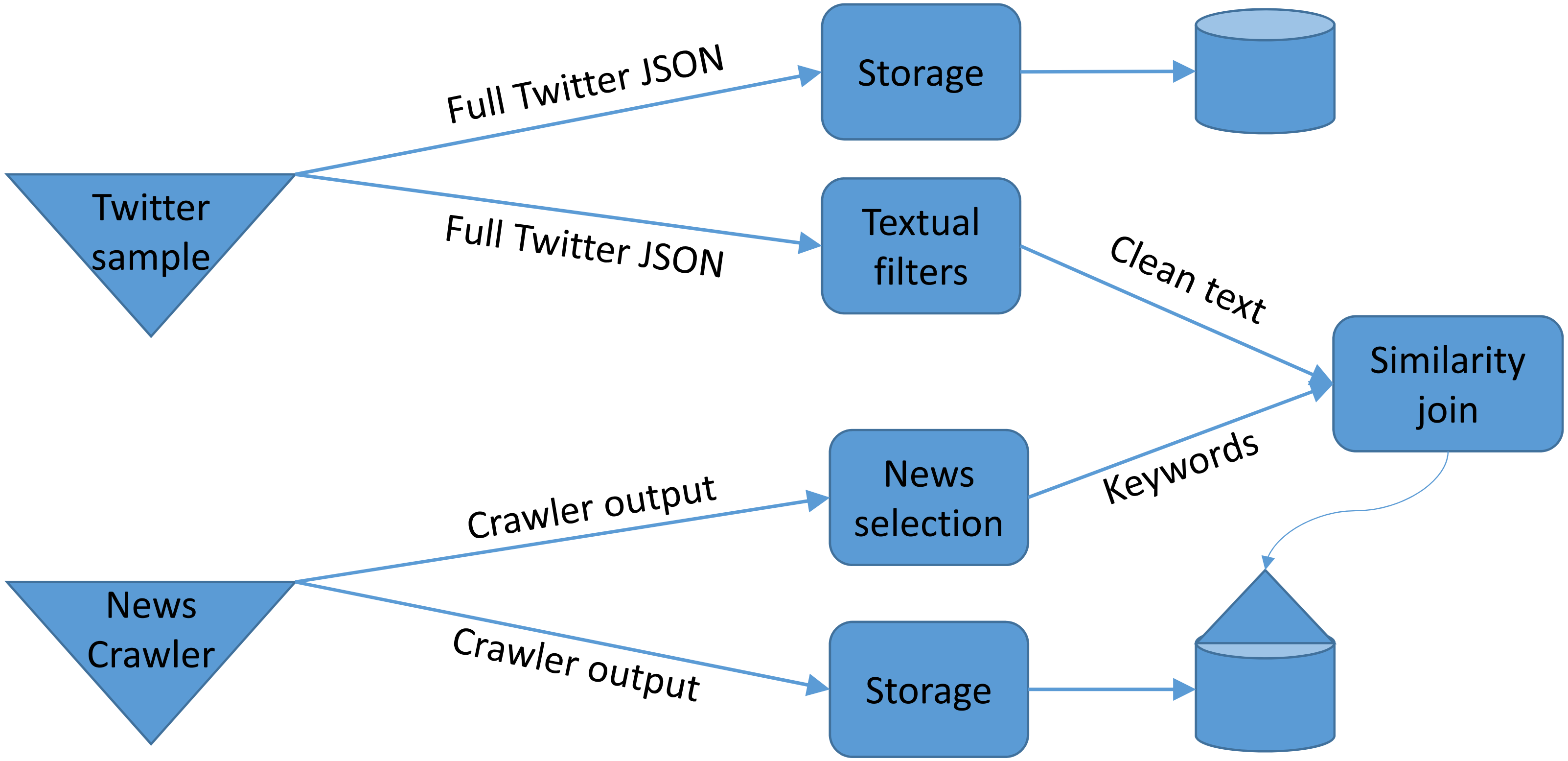
19.5 19.6 19.7	19.8 19.9 20.0	20.1 20.0 20.1	20.1 20.1 20.2	20.2 20.3 20.4	20.5 20.5 20.4 20.5 20.5

Processing example

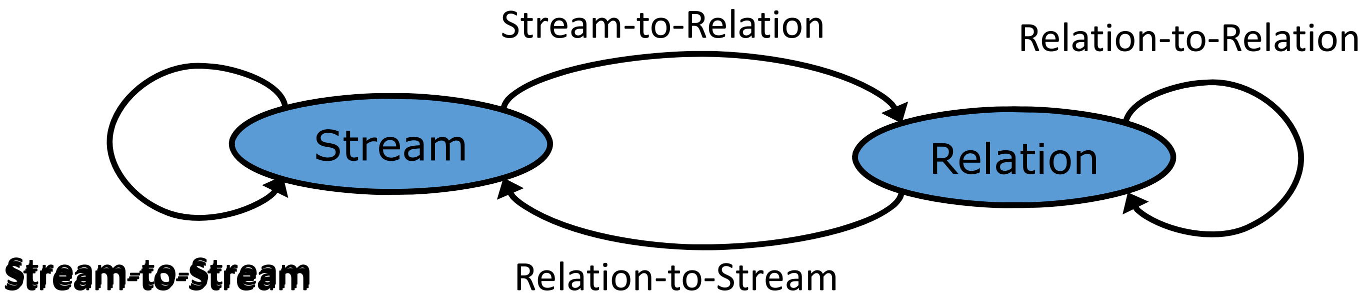


Binary operations

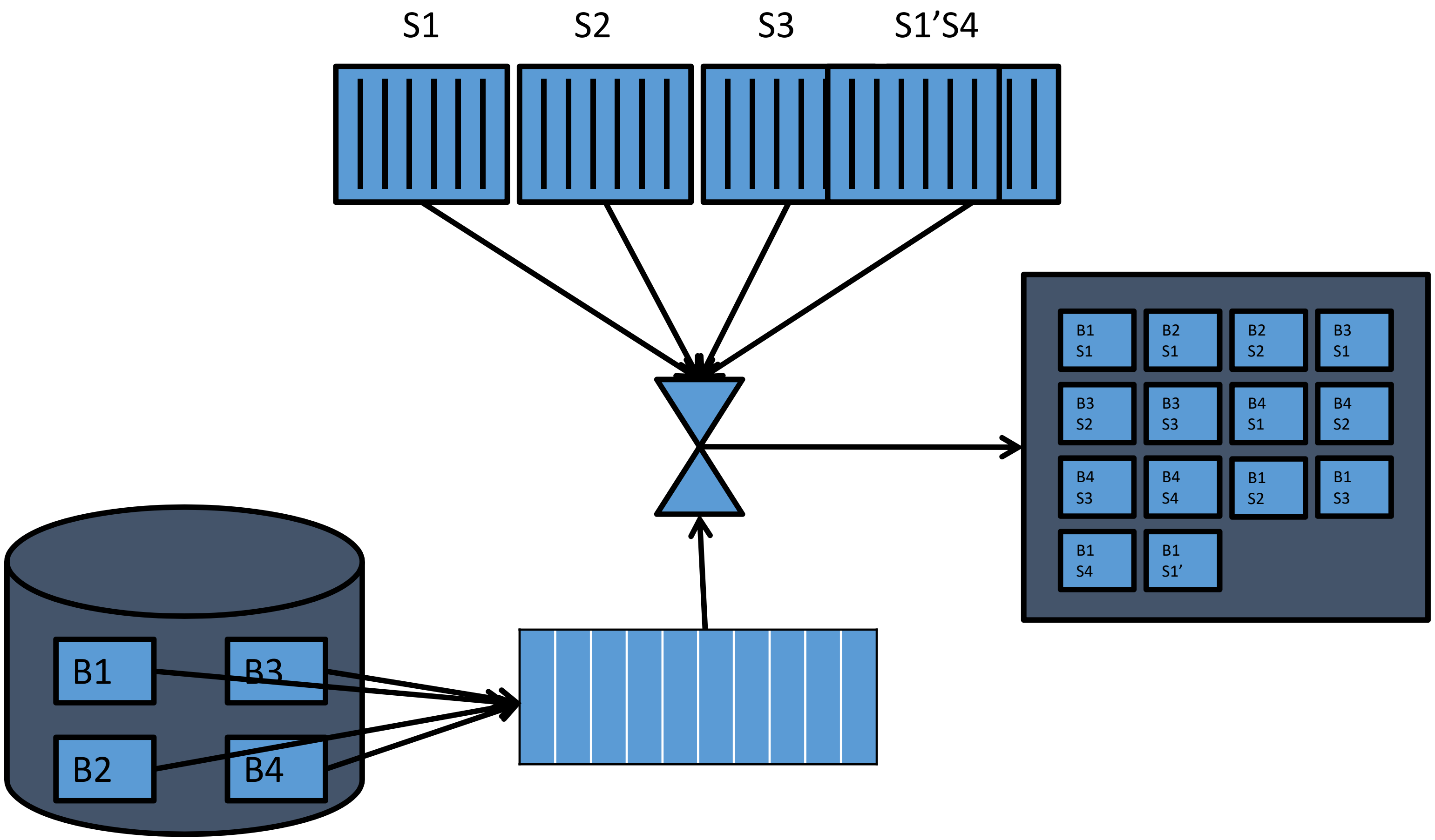
Stream-to-Stream



Kinds of binary operations



Meshjoin algorithm example



Meshjoin algorithm

- Algorithm

```
while true do
    read next block of R into a memory page
    if queue is full then
            dequeue w tuples
    endif
    foreach t in queue do
            generate t join R (for the current block)
    endforeach
endwhile
```

D = Time to access one block
C = Time to process one tuple
B = Blocks of R
R_S= Stream tuples per page
w = Stream tuples removed per loop
λ = Arrival-stream rate (tuples/sec)
μ = Service-join rate (tuples/sec)

- Cost of one loop (assuming M+2 memory pages)
  - $D+M \cdot R_S \cdot C = D+w \cdot B \cdot C$
- Considerations
  - Performs a cyclic scan of the table and keeps a sliding window of the stream
  - We try to maximize μ ($\lambda \leq \mu$) given M
    - $w = M \cdot R_S/B \Rightarrow M = B \cdot w/R_S$
    - $\mu = w/(D+w \cdot B \cdot C) = 1/(D/w+B \cdot C)$
  - This would often be faster than Row Nested Loops
    - $\mu = 1/(h \cdot D+D)$

Stream processing

Relational temporary tables

CREATE GLOBAL TEMPORARY TABLE <tablename> (…)
[ON COMMIT {<u>DELETE ROWS</u>|PRESERVE ROWS}];

- Relational mapping
  - Each element is a tuple
  - The sliding window is a relation
- Data is not persistent
  a) Transaction specific
  b) Session specific
- Does **not** support:
  - Foreign keys
  - Cluster
  - Partitions
  - Parallelism

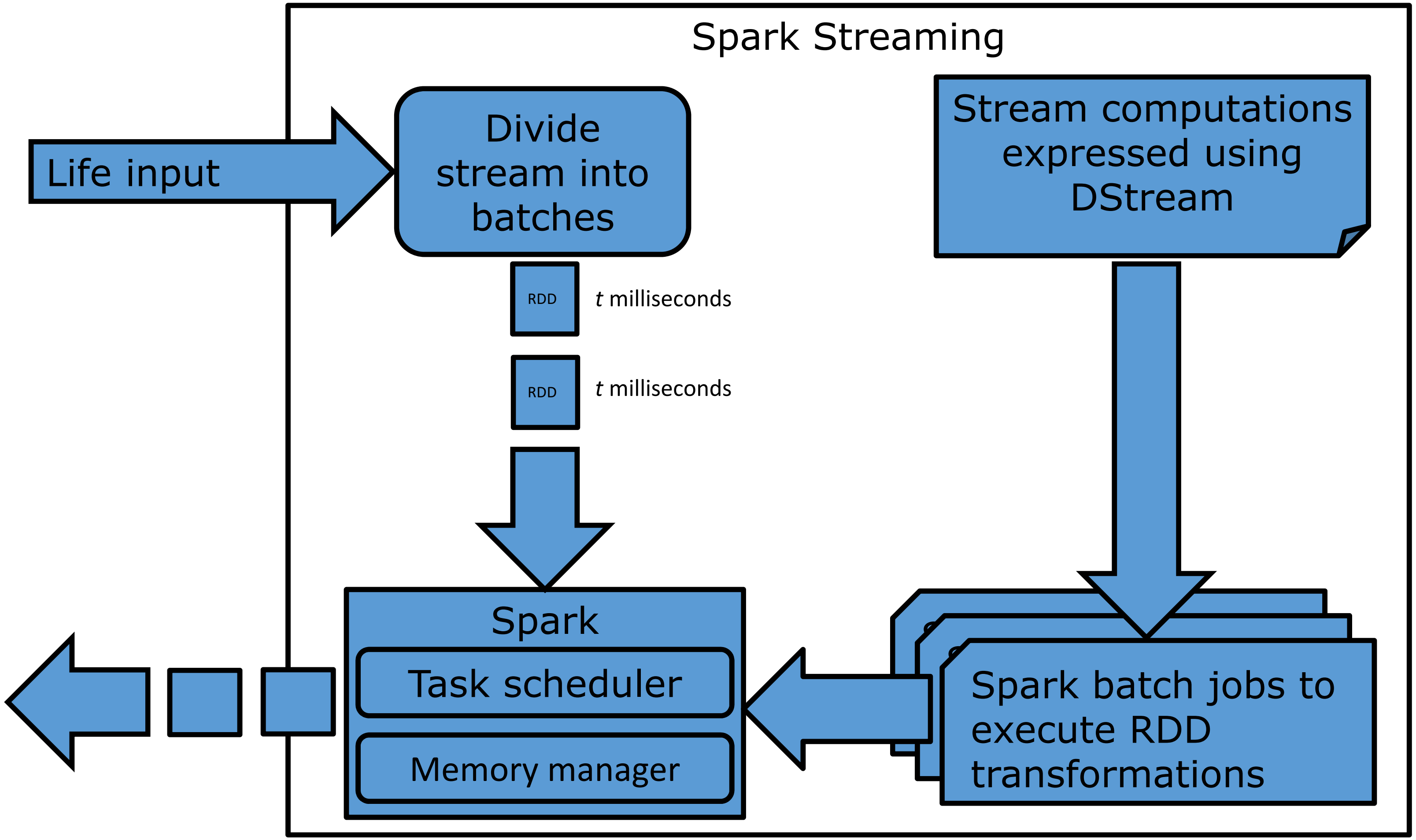
Databases vs Streams

	Database management	Stream management
Data	Persistent	Volatile
Access	Random	Sequential
Queries	One-time	Continuous
Support	Unlimited disk	Limited RAM
Order	Current state	Sorted
Ingestion rate	Relatively low	Extremely high
Temporal requirements	Little	Near-real time
Accuracy	Exact data	Imprecise data
Heterogeneity	Structured data	Imperfections
Algorithms	Multiple passes	One pass

Spark streaming goals

- Scalability to hundreds of nodes
- Minimal overhead
  - Sub-second latency
- Recovery from faults and stragglers
  - On reception, data is replicated to a second executor in another worker
  - State (i.e., summary) is periodically (e.g., every 10 RDDs) saved to a reliable file system

Discretized Stream (micro-batches)



Transformations vs Output operations

- Transformations
  - Stateless (depend on current RDD)
    - Same as in Spark
  - Stateful (depend on past RDD)
    - window(windowDur, slidingDur)
    - reduceByWindow(windowDur, slidingDur, aggregation)
    - updateStateByKey(function)
    - mapWithState(function)
- Actions
  - save
  - foreachRDD(sparkCode)

Note: Both windowDuration and slidingDuration must be multiples of batchDuration

D-Stream example

```
// Create a StreamingContext with a 1-second batch size from a SparkConf
JavaStreamingContext jssc = new JavaStreamingContext(conf, Durations.seconds(1));
// Create a DStream from all the input on port 7777
JavaDStream<String> lines = jssc.socketTextStream("localhost", 7777);
// Filter our DStream for lines with "error"
JavaDStream<String> errorLines = lines.filter(new Function<String, Boolean>() {
public Boolean call(String line) {
return line.contains("error");
}});
// Print out the lines with errors
errorLines.print();
// Start our streaming context and wait for it to "finish"
jssc.start();
// Wait for the job to finish
jssc.awaitTermination();
```

Architectural patterns for stream/event processing

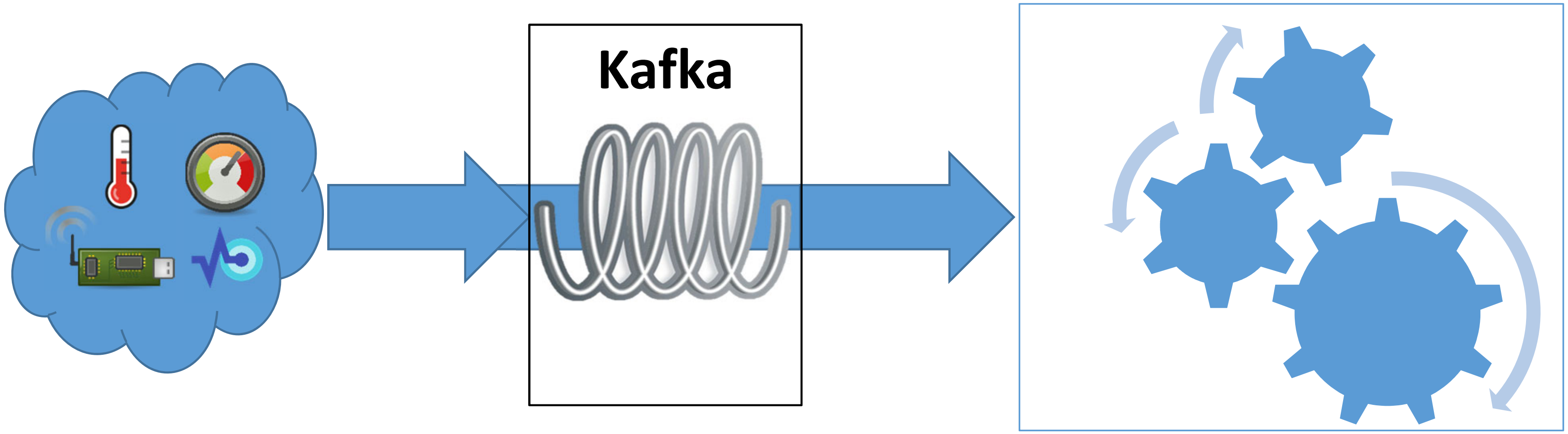
Architectural patterns

A. Stream ingestion
B. **Near**-real time
   - Non-partitioned
     - Get profile information (lookup) needed for decisions
     - Requires nearly no coding beyond the application-specific logic
   - Partitioned
     - Define a key to partition data
       - Match incoming data to the subset of the context data that is relevant to it
C. Pattern matching
D. Complex topology
   - Aggregation
   - Machine learning

https://blog.cloudera.com/architectural-patterns-for-near-real-time-data-processing-with-apache-hadoop

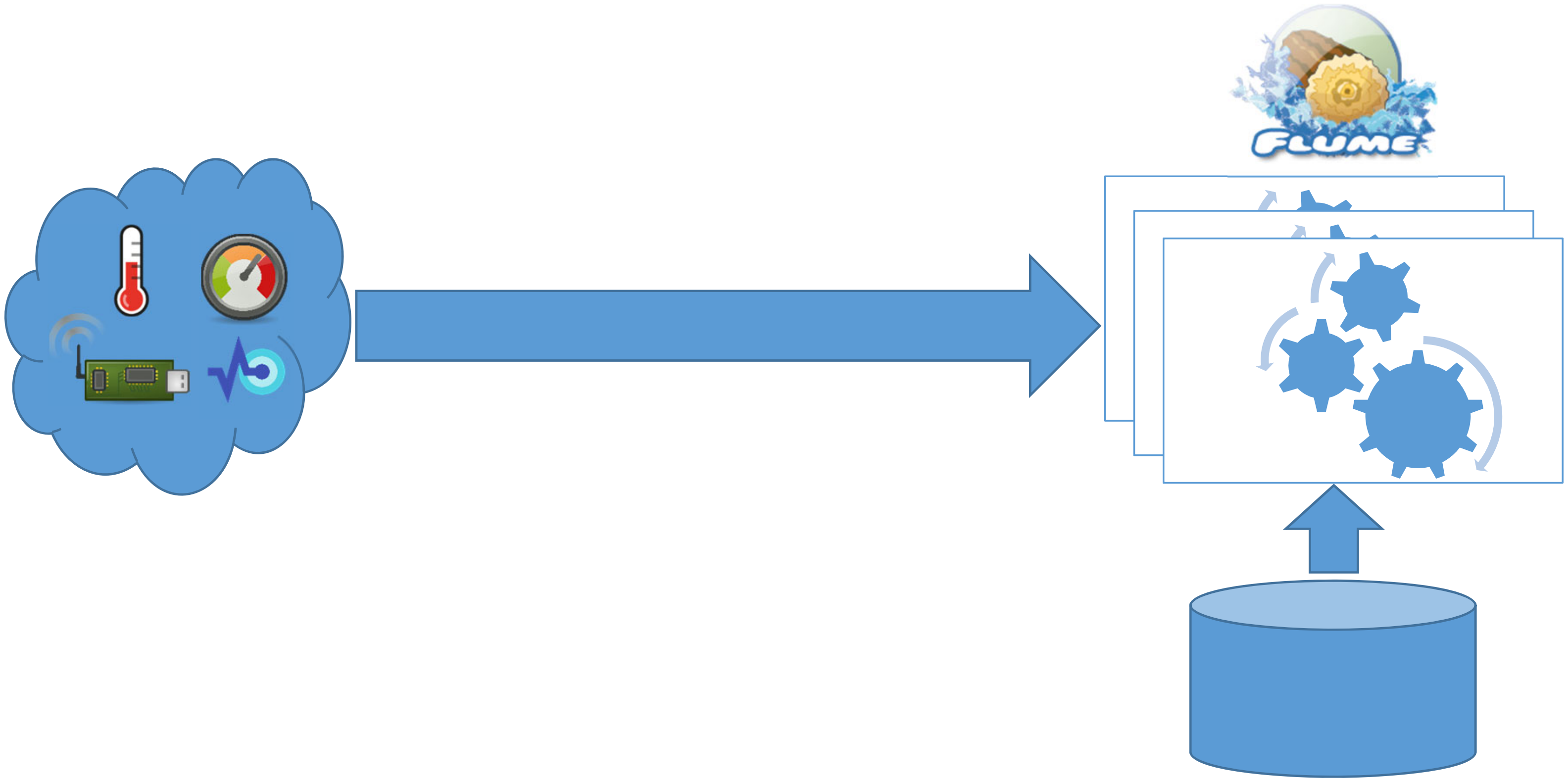
A. Stream ingestion

- The objective is to not lose any event



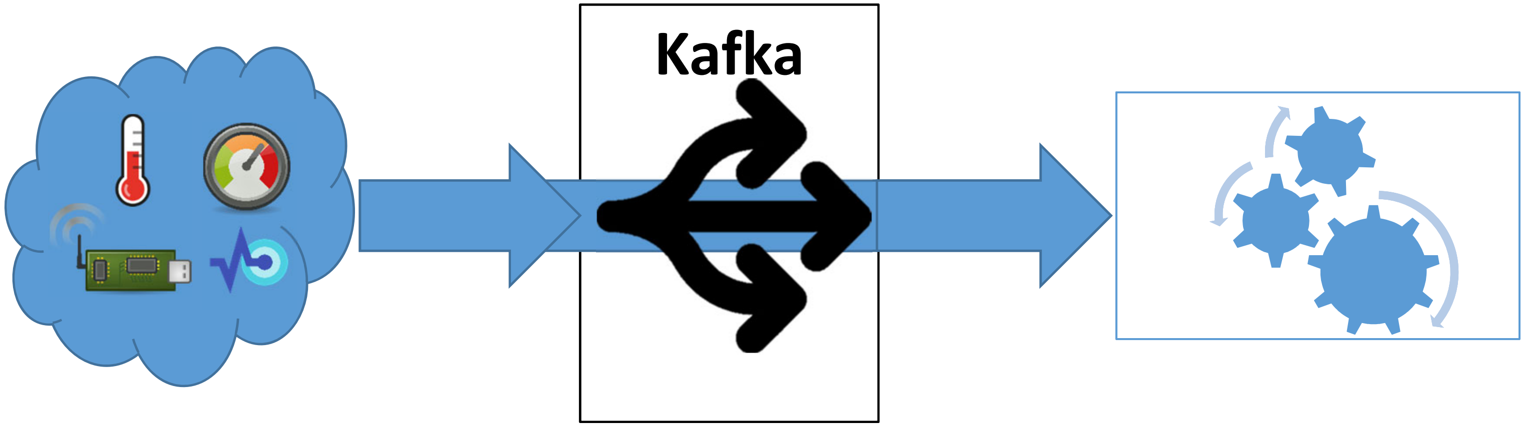
B. Near-real time event processing (I)

- The objective is to react as soon as possible



B. Near-real time event processing (II)

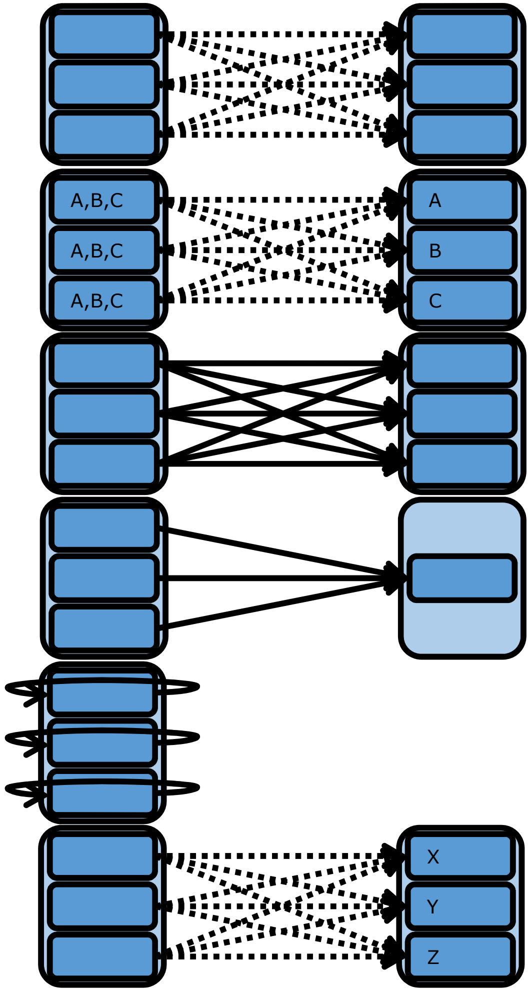
- The objective is to react as soon as possible



C. Complex Event Processing

- Pattern matching
  - State keeps all potential matches
    - Tree
    - NFA (Non-deterministic Finite Automata)
- Hard to distribute
- Consider
  - Time constraints
  - Absence of events
  - Re-emitting complex events

D. Complex topology



- Shuffle grouping
  - Random
- Fields grouping
  - Same value, same task
- All grouping
  - Broadcast to all task
- Global grouping
  - All data converges to one task
- None grouping
  - Execution stays in the same thread (if possible)
- Direct grouping
  - Producers direct the output to a concrete task

Closing

Summary

- Stream definition and characterization
  - Complex event processing
- Streaming architectural patterns
- Streaming operations
  - Sliding windows
  - Binary operations
- Spark streaming
  - Architecture

References

- M. Stonebraker et al. *The 8 Requirements of Real-Time Stream Processing*. SIGMOD Record 34(4), 2005
- N. Polyzotis et al. *Meshing Streaming Updates with Persistent Data in an Active Data Warehouse*. IEEE Trans. Knowl. Data Eng. 20(7), 2008
- L. Liu and M.T. Özsu (Eds.). Encyclopedia of Database Systems. Springer, 2009
- I. Botan et al. *Flexible and Scalable Storage Management for Data-intensive Stream Processing*. EDBT, 2009
- T. Akidau et al. *The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in MassiveScale, Unbounded, OutofOrder Data Processing*. VLDB, 2015
- I. Flouris. *Issues in complex event processing: Status and prospects in the Big Data era*. J. of Systems and Software 127, 2017