RealTime System Architecture

Lecture-7

Introduction

- Real-time systems are defined as those systems in which the correctness of the system depends not only on the logical result of computation, but also on the time at which the results are produced.

"A real time system has performance deadlines on its computations and actions."

- **Deadline:** time when execution must be completed. A deadline is either a point in time (time-driven) or a delta-time interval (event- driven) by which a system action must occur.

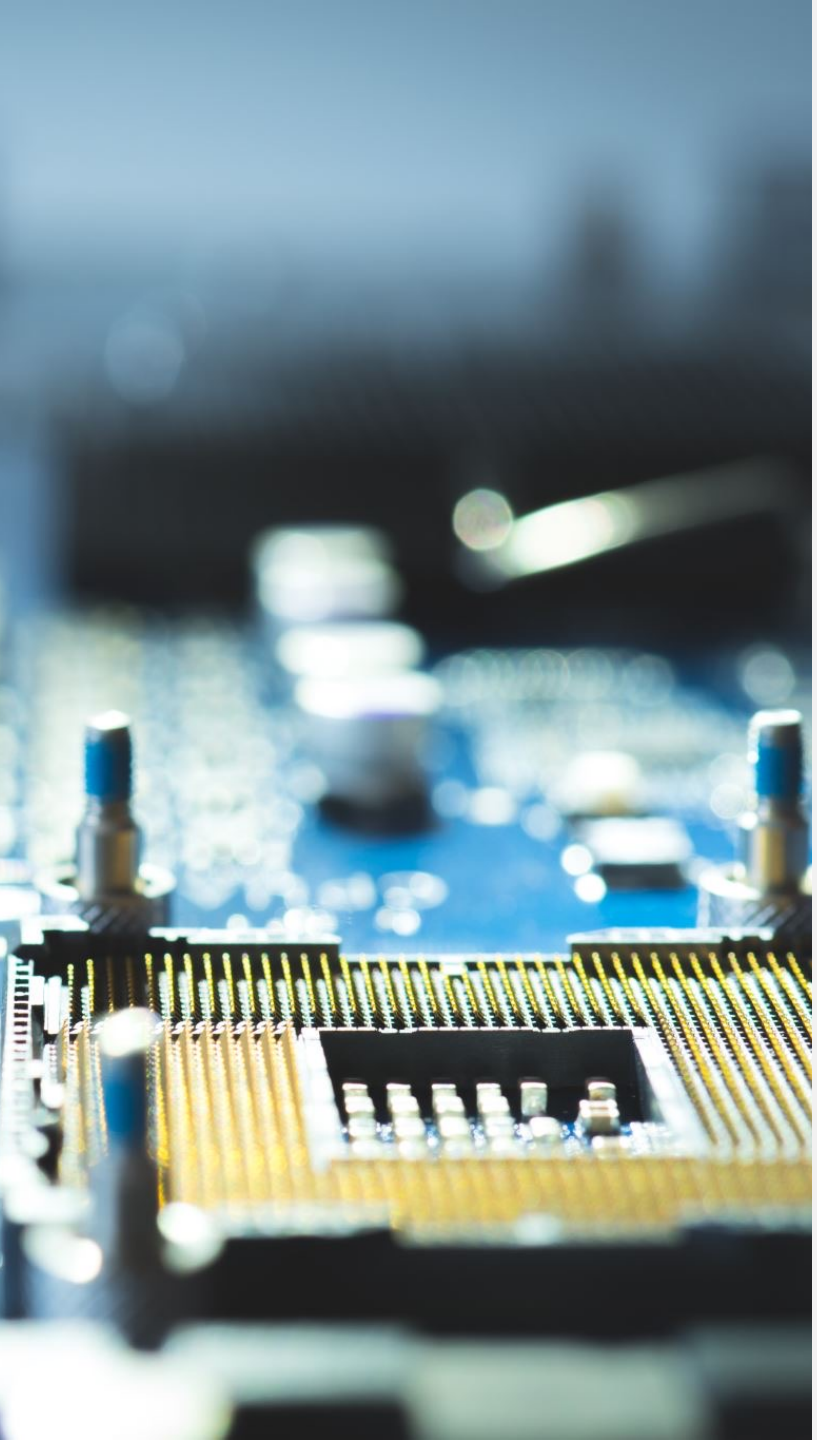
  Hard deadline: late data may be a bad data.

  Soft deadline: late data may be a good data.

Hard real-time systems (e.g., Avionic control).

Firm real-time systems (e.g., Banking).

Soft real-time systems (e.g., Video on demand).



Applications

- Industrial Applications (Chemical Plant Control, Automated Car Assembly Plant)
- Medical (Robot used in recovery of displaced radioactive material)
- Peripheral Equipment (Laser Printer, Scanner)
- Automotive and Transportation (Multi Point fuel injection System)
- Telecommunication application (Cellular Systems)
- Aerospace (Computer on Board an aircraft)
- Internet and Multimedia (Video Conferencing)
- Defence applications (Missile Guidance System)
- Miscellaneous Applications (Railway Reservation System)

Characteristics of RTS

- Time constraints (dead line associated with tasks)
- New Correctness Criterion (not only logical correctness, time is also important)
- Safety – Criticality (failure of the system cause extensive damages)
- Concurrency (RTS must process concurrent data)
- Distributed and feedback structure
- Task Criticality (is a measure of the cost of failure of a task)
- Reactive (RTS are often reactive means an ongoing interaction between system and environment is maintained)
- Stability
- Exception handling

Safety & Reliability

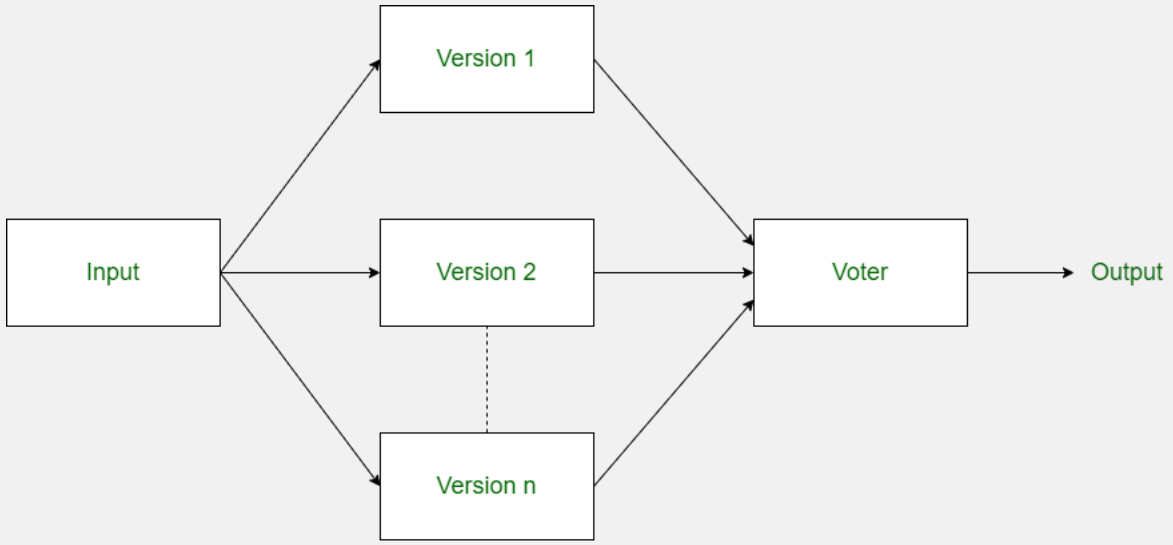
- In Traditional system safety and reliability are independent issues
- In RTS safety and reliability are dependent issues
- **Fail safe state:** of a system is one which if entered when the system fails, no damage would result.
- **Safety critical systems:** is one whose failure can cause severe damages

How to achieve high reliability?

- Error avoidance
- Error detection and removal
- Fault tolerance

Software Fault Tolerance

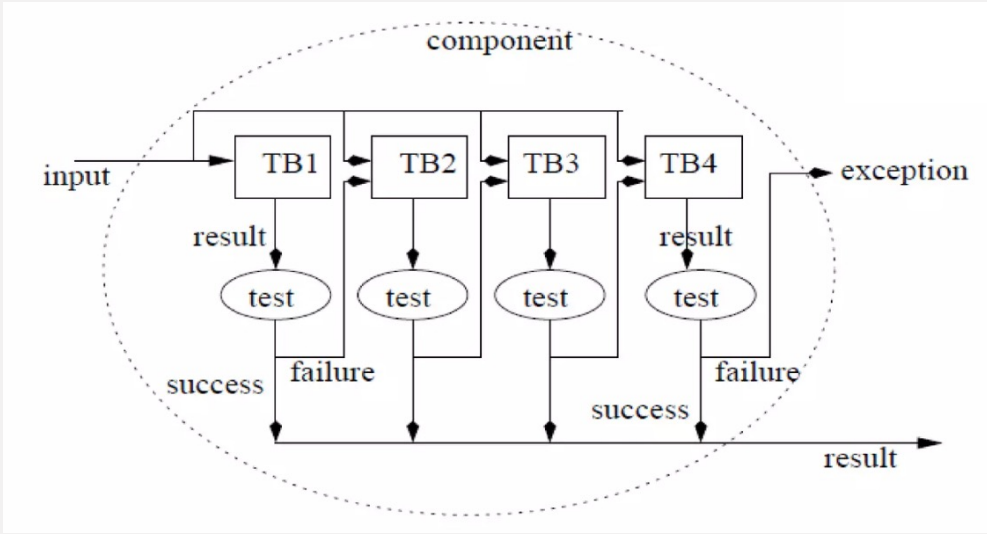
- **N – version programming**
  - ➢ *An adaption of TMR technique*
  - ➢ *Is not very successful in achieving fault tolerance*



N-version Programming

- **Recovery Block**

  In recovery block technique, different algorithms are used for different try blocks. Try blocks are basically the redundant components. Here the redundant copies are not run simultaneously. The result in each try is block is tested by acceptance test.



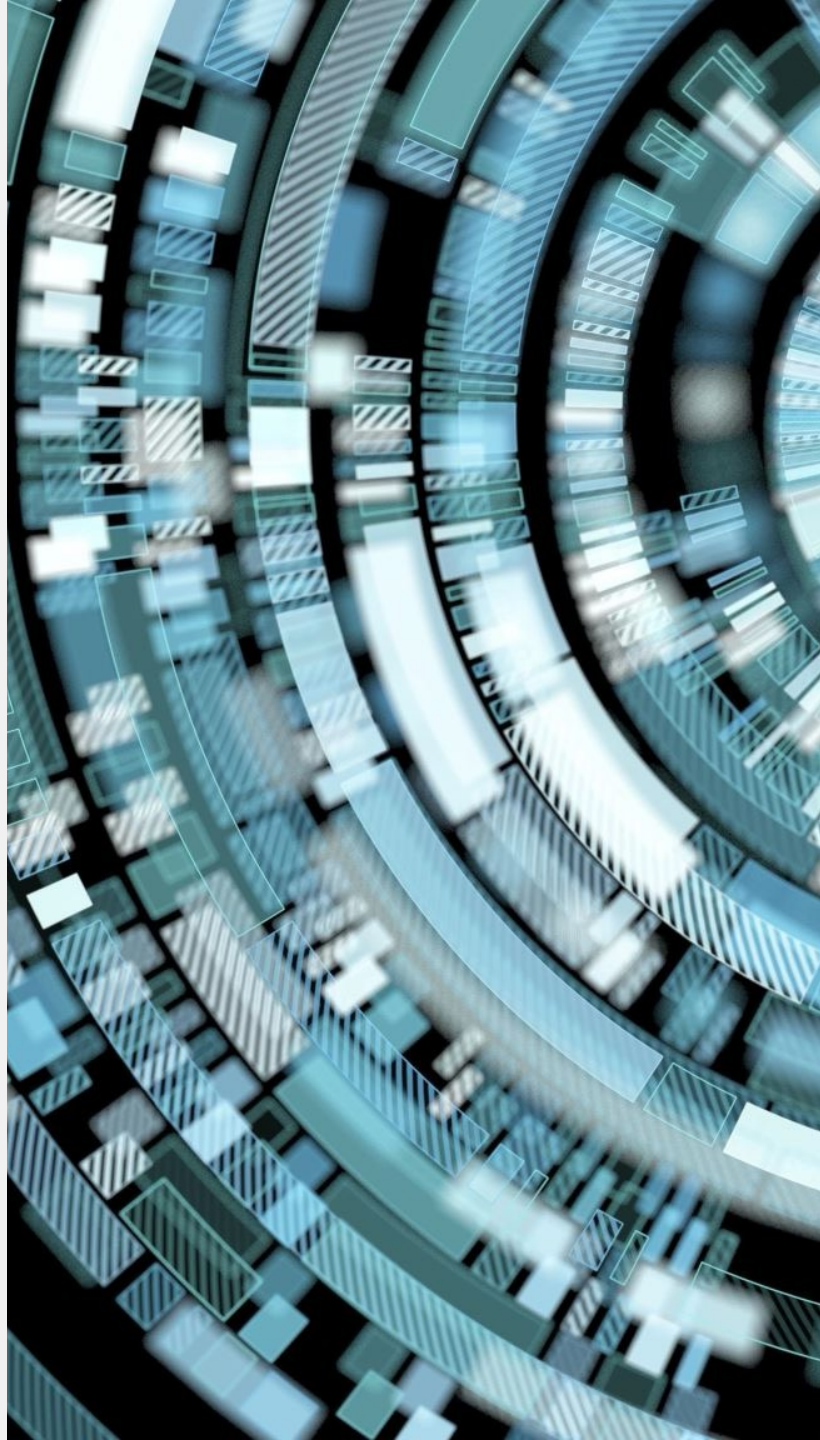
Recovery Block

Difference b/w N-Version & Recovery Blocks

N-VERSION PROGRAMMING	RECOVERY BLOCKS
N-versions of software are developed by n independent team.	Redundant copies are developed using different algorithms.
Redundant copies are run concurrently.	Redundant copies are run one by one.
Acceptance test is not performed.	Acceptance test is performed.
It can be applied top critical systems.	It can't be applied to critical systems.
It is possible to achieve same fault for different version.	Same fault can't be achieved by redundant copies.
It has statistical correlation of failure.	It is used only if task deadline is more than task computation time.
It can be used with task having more or less laxity.	It is used with tasks having more laxity.

Data Stream Processing

- Market Trends
- Overview of Data Streams
- Overview of Stream Processing
- Cloud computing and stream processing



Market Trends

It's time to adopt Stream Data Integration

- The global streaming analytics market size is expected to grow from USD 12.5 billion in 2020 to **USD 38.6 billion by 2025**.
- More than half of major new business systems will incorporate **real-time data** and **continuous intelligence** to improve decisions.
- More and more organizations will use **streaming data** to support their data integration use cases.

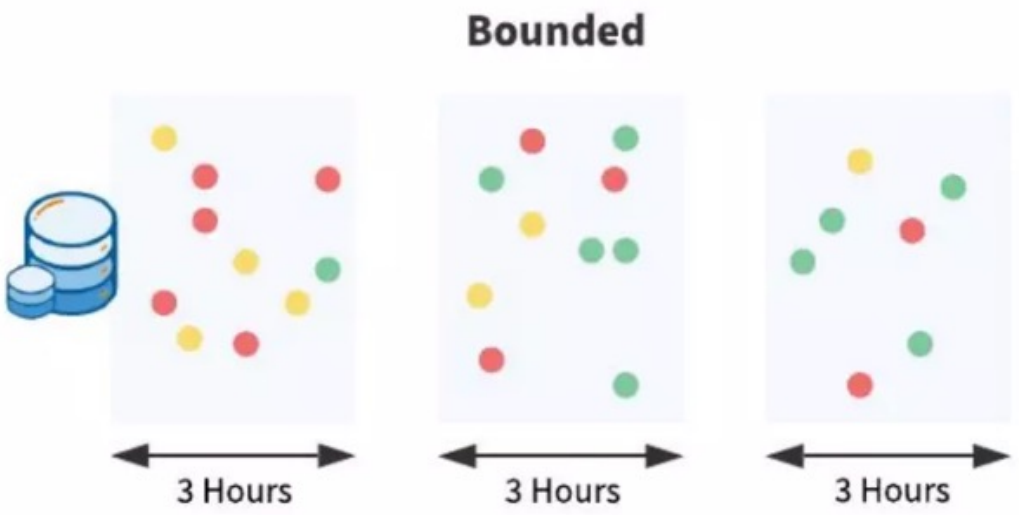
Market Trends

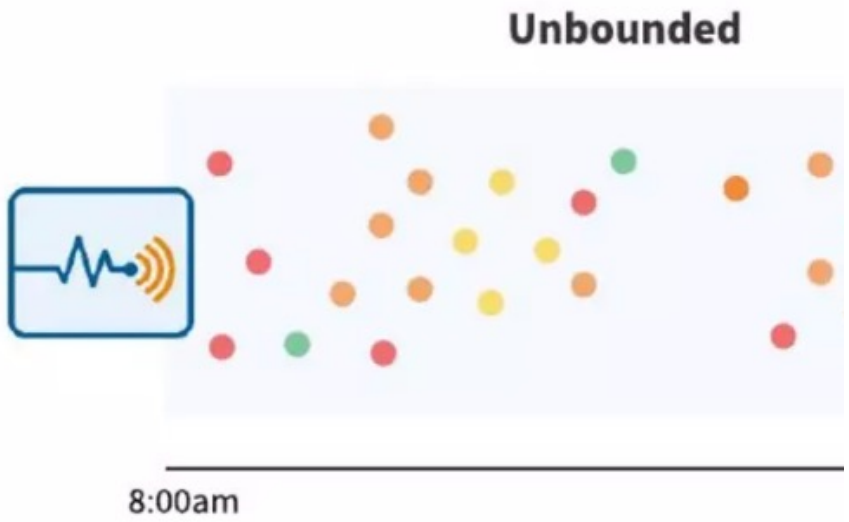
Why are more businesses leveraging stream processing?

- Explosive growth of Sensors and hardware that collect data.
- Technology to capture, store, and process data continues to improve.
- Actionable information delivered faster to decision makers is a competitive advantage.

Data Streams

- **Bounded data** is finite and has a discrete beginning and end. It is associated with batch processing.
- **Unbounded data** is infinite, having no discrete beginning or end. It is associated with stream processing.

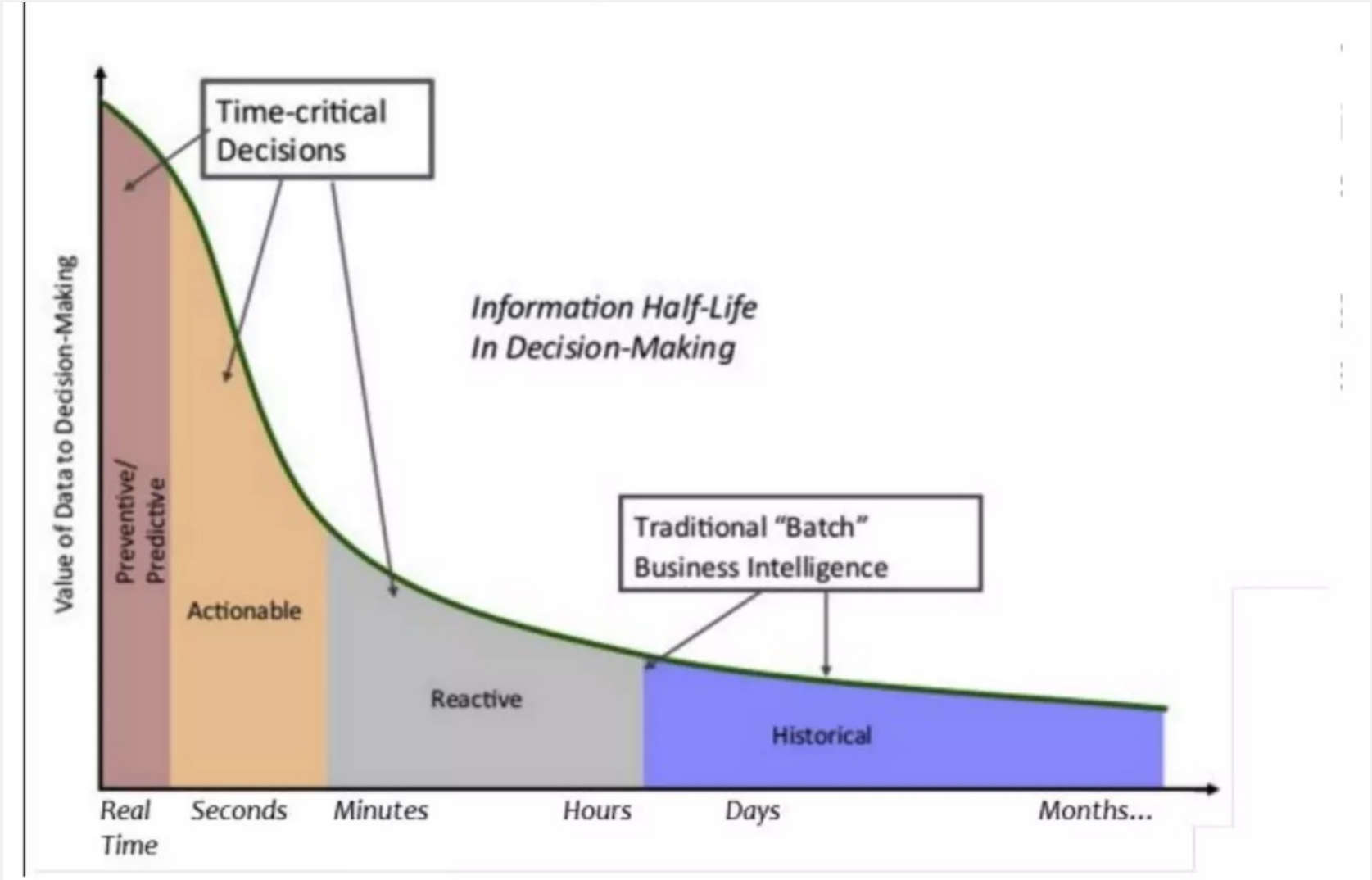




Characteristic of Data Stream

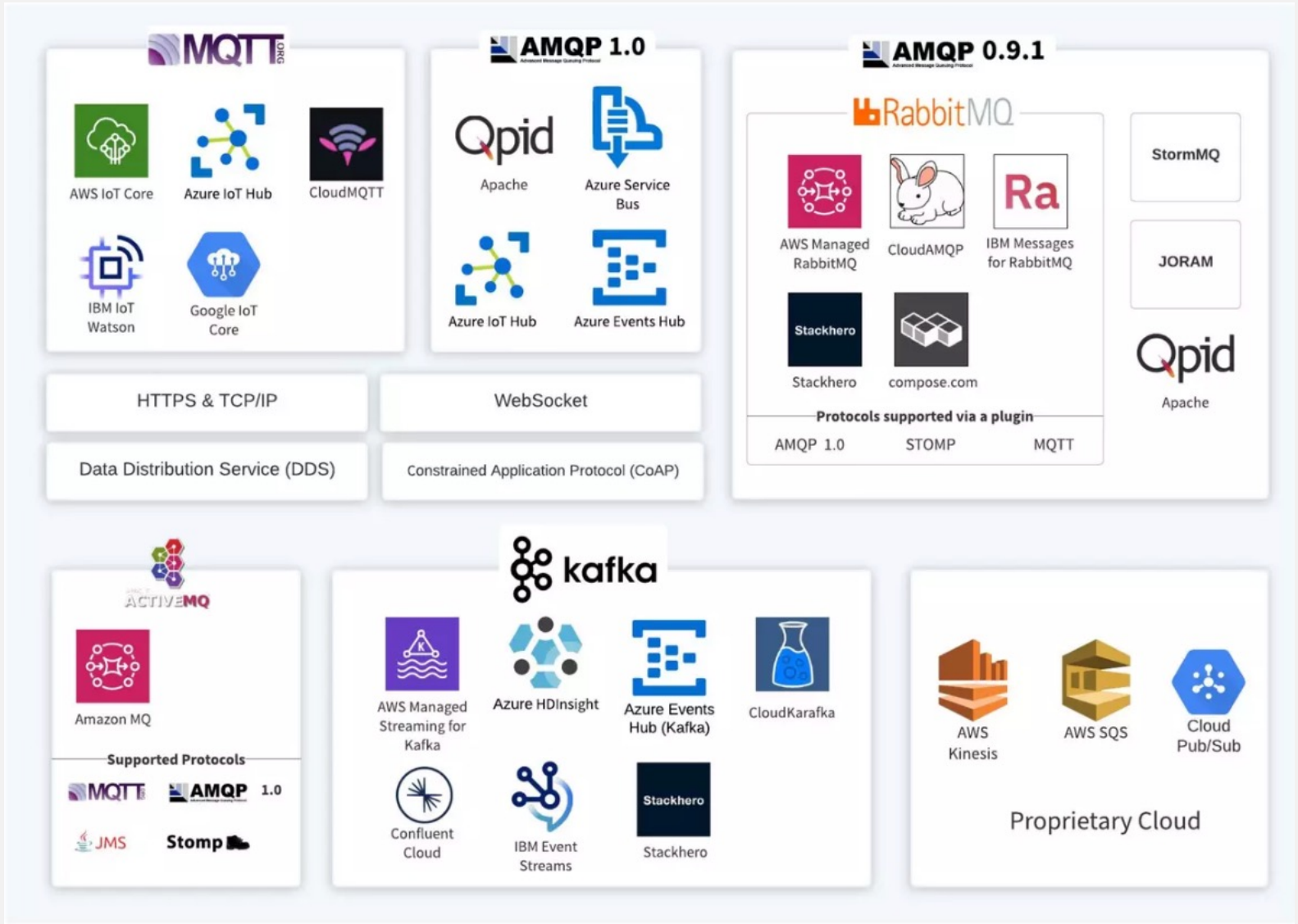
- Data records are small in size.
- Data volumes can be extremely high.
- Data distribution can be inconsistent with quiet and busy periods.
- Data can arrive out of sequence compared to when the event happened.

Diminishing Value of Data



Source: Amazon Web Services

Data Streaming Technologies



Stream Processing

Stream processing is a term that groups together the collection, integration, and analysis of unbounded data.

“Stream Processing delivers insights to organizations on a continuous basis.”

High & Low Volume Message Streams

There are three ways organization work with unbounded data

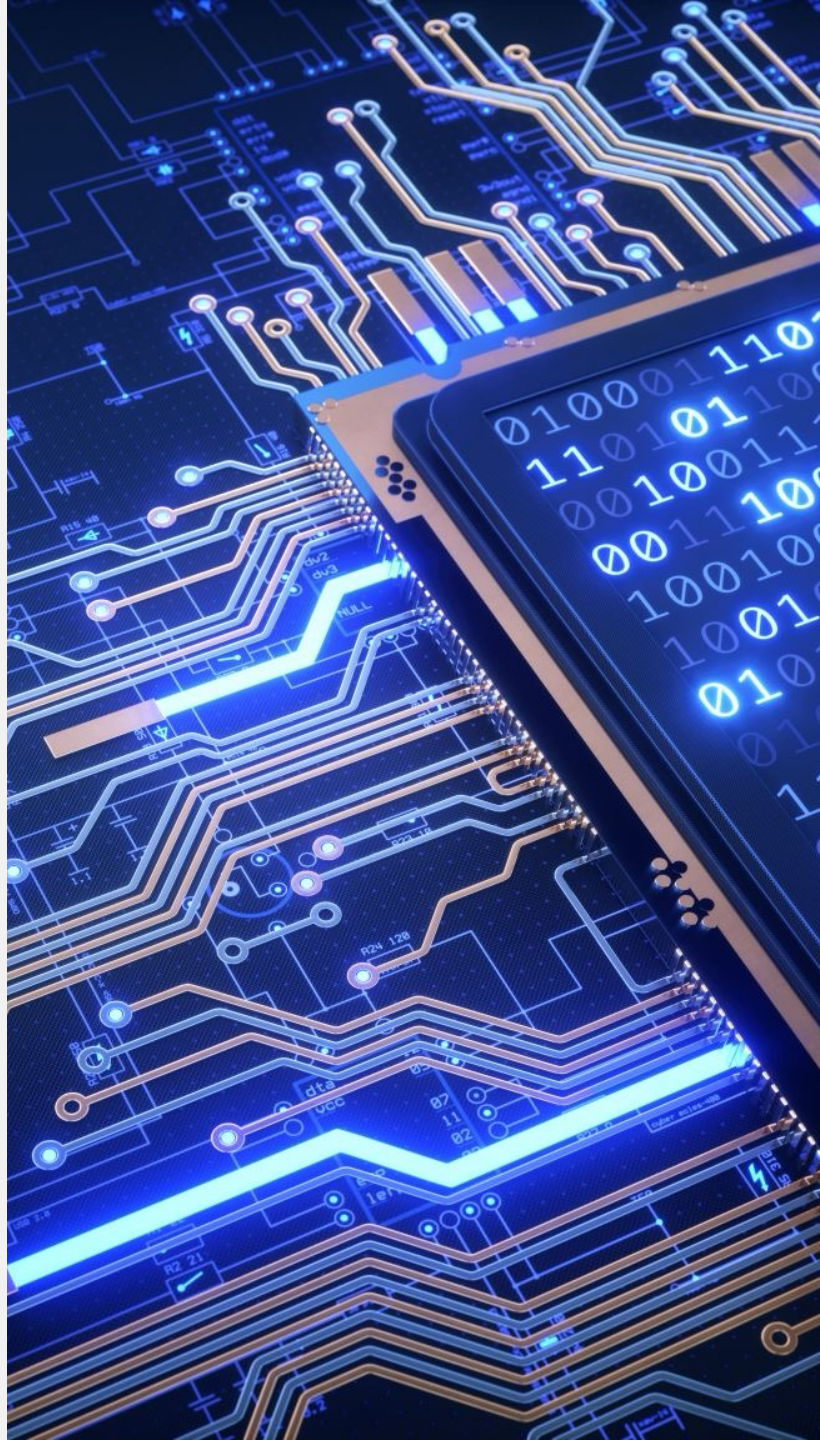
- **Batch processing of stored data** – Unbounded data is stored and processed at a specified interval → ***[Near Real Time]***
- **Event Processing** – Each event in the unbounded stream is handled separately with connections between events being stored in persistent storage → ***[Real Time Low Velocity]***
- **Stream Processing** – Continous ***processing*** of high-volume data streams. Data is processed in memory before storing. → ***[Real Time Any Velocity]***

Batch Processing vs Stream Processing

	Batch Processing	Stream Processing
Frequency	Infrequent jobs that produce results once the job has finished running.	Continuously running jobs that produce constant results.
Performance	High latency (minutes to hours).	Low latency (seconds to milliseconds).
Data Sources	Databases, APIs, and static files.	Message queues, event streams, and transactions.
Analysis Type	Complex analysis is possible.	Simpler analysis, including aggregation, filtering, enrichment, proximity analysis, and event detection.
Processing	Process after storing the data.	Process and then maybe store the data.

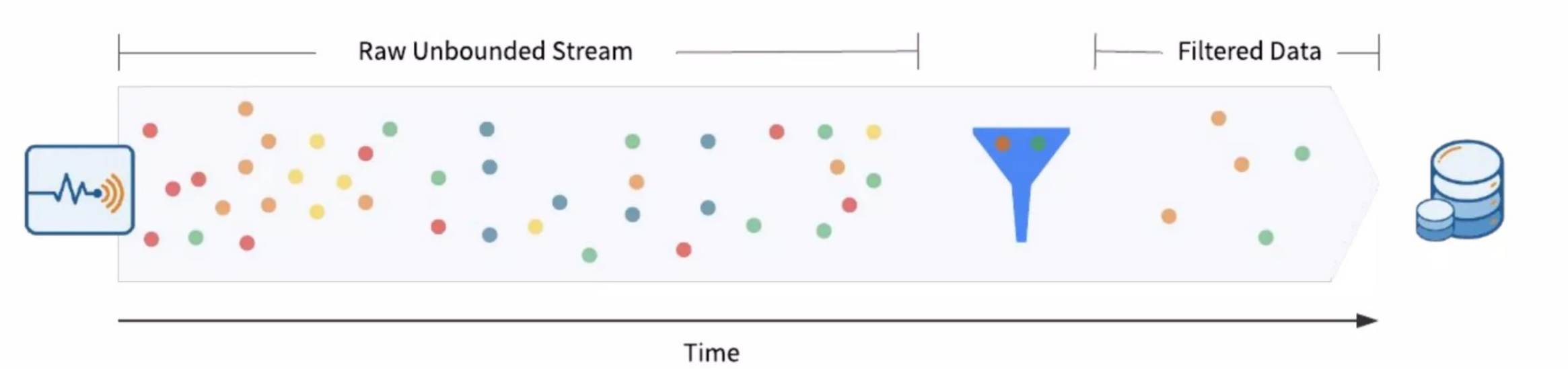
Core Stream Processing Workflows

- Filtering
- Enriching
- Aggregating
- Event Detection



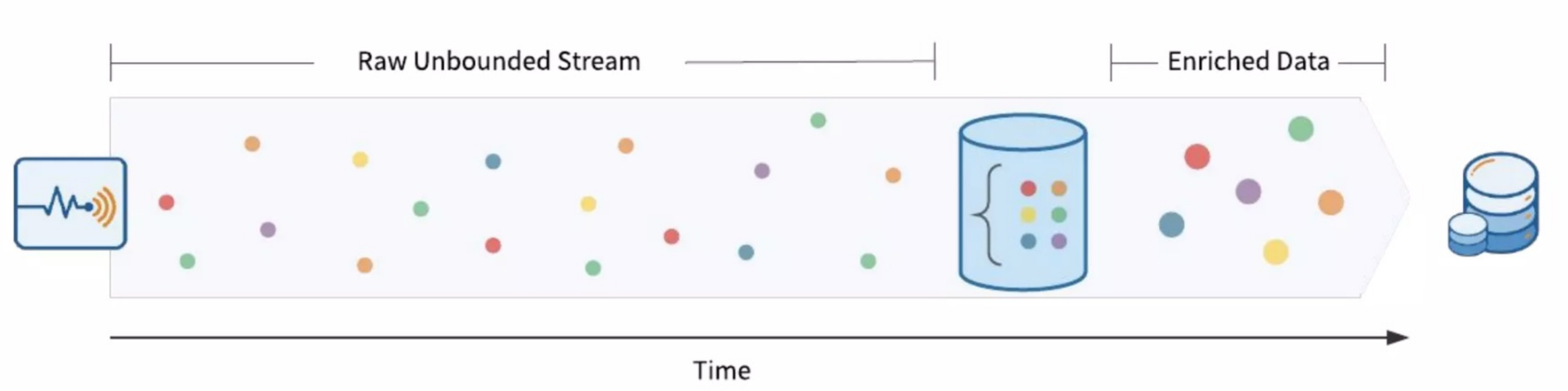
Filtering

Reduced data volume in memory before committing data to disk



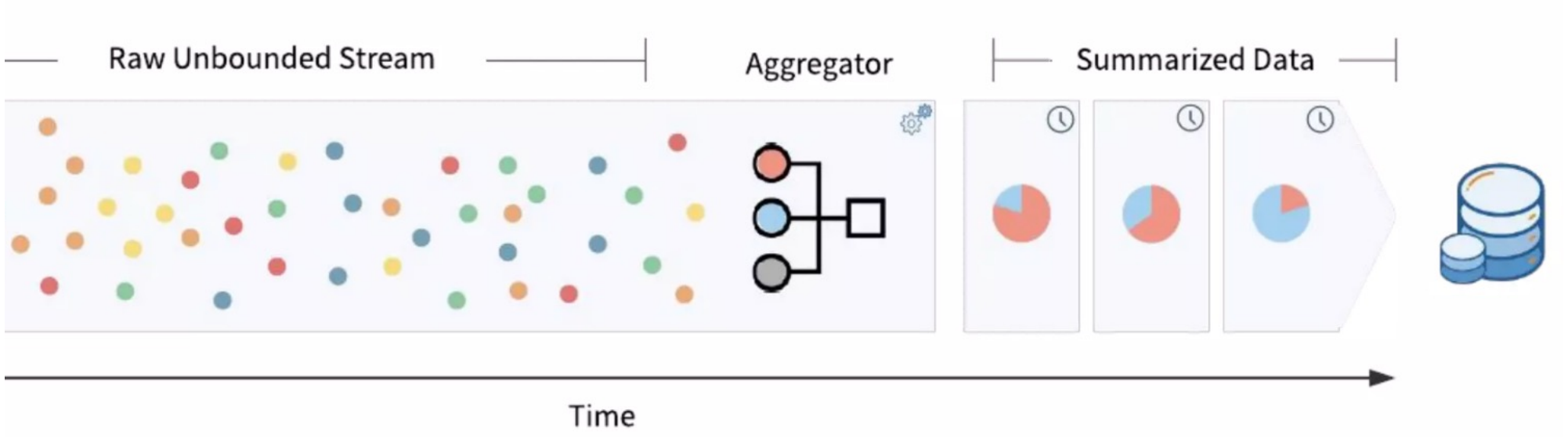
Enriching

Join the unbounded data to other dataset (databases, APIs) before committing data to disk



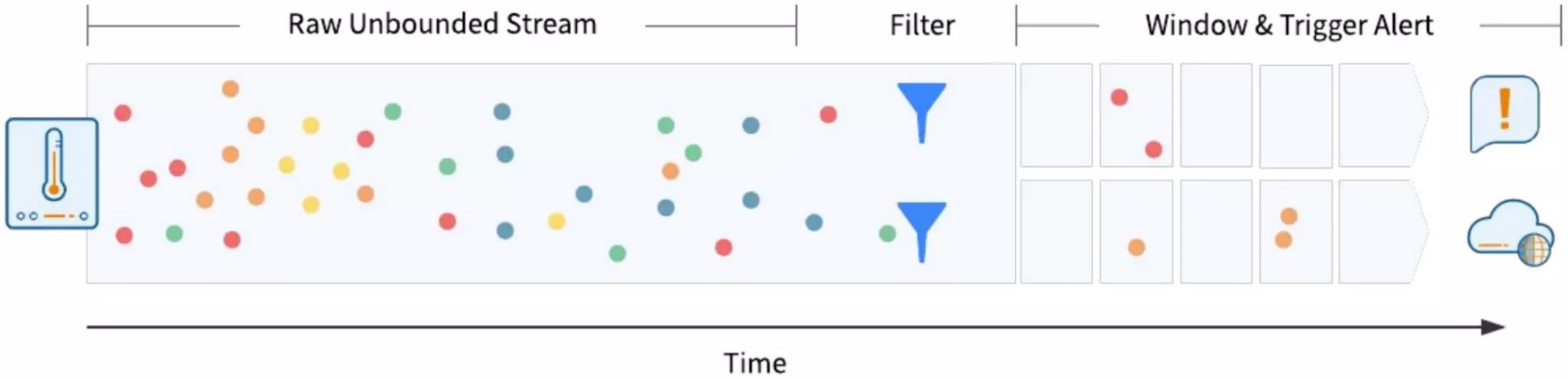
Aggregating

Summarize the unbounded data by calculating time windowed aggregations before committing data to disk



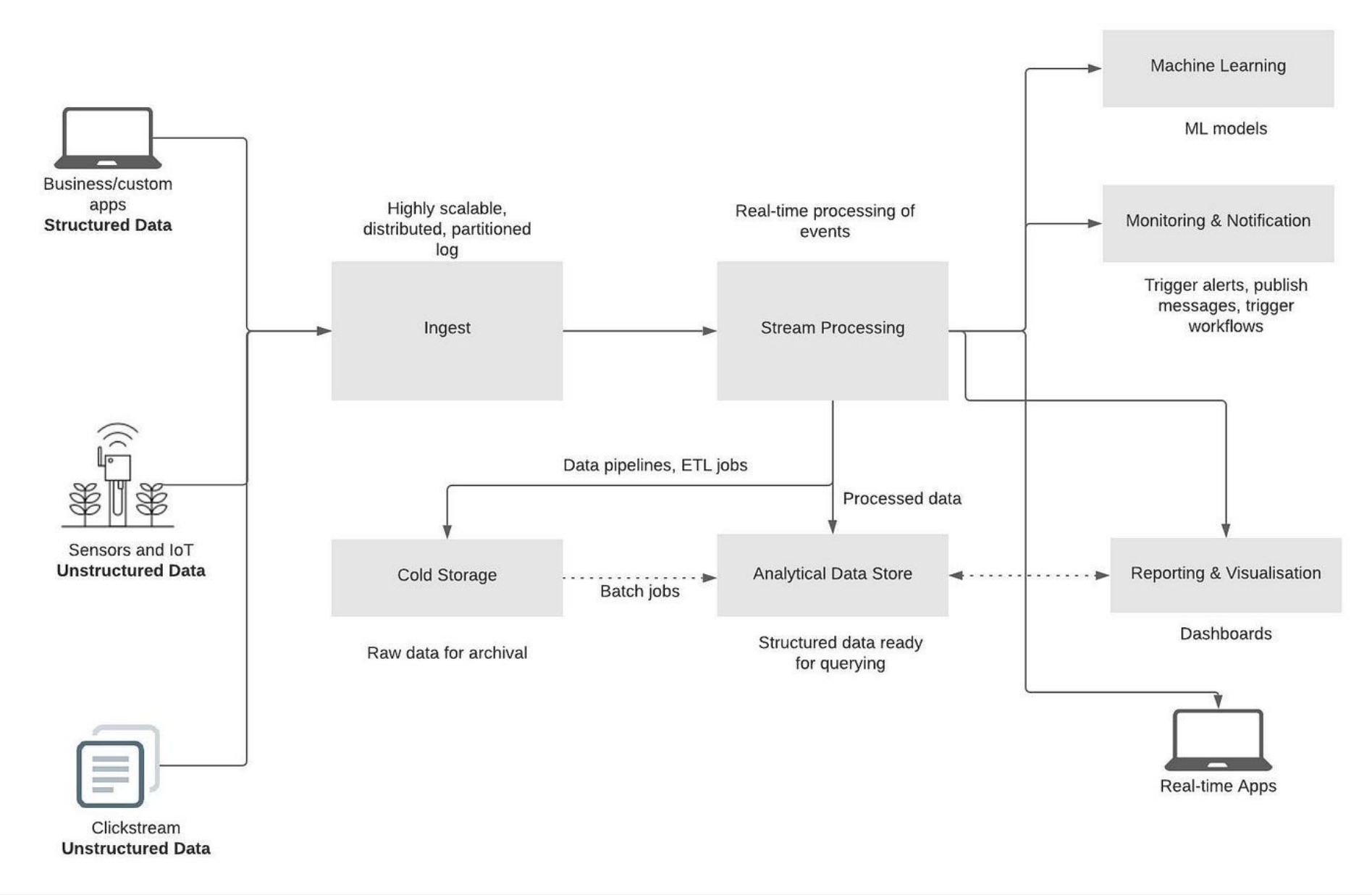
Event Detection

Detect patterns in memory and then trigger an event when certain criteria are met



Cloud Services Related to Stream Processing

- Data Warehouses
- Data lakes
- Stream Processing
- IoT Device Connection
- Machine Learning
- Scalable Compute



Logical component of stream processing architecture