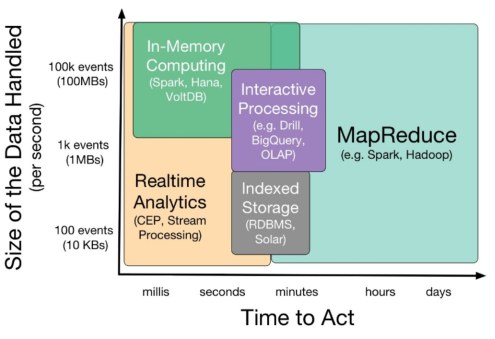
**Streamıng Analytıcs Pattern**

Stream processing technologies like Apache Samza and Apache Storm has received much attention under the theme large scale streaming analytics. However, these tools force every programmer to design and implement real-time analytics processing from first principals.

For an example, if users need a time window, they need to implement it from first principals. This is like every programmer implementing his own list data structure. Better understanding of common patterns will let us understand the domain better and build tools that handle those scenarios. This tutorial tries to address this gap by describing 13 common real-time streaming analytics patterns and how to implement them. In the discussion, we will draw heavily from real life use cases done under Complex Event Processing and other technologies



**Realtime Streaming Analytics Patterns**

Before looking at the patterns, let’s first agree on the terminology. Realtime Streaming Analytics accepts input as a set of streams where each stream consists of many events ordered in time. Each event has many attributes, but all events in a same stream have the same set of attributes or schema.

**Pattern 1: Preprocessing**

Preprocessing is often done as a projection from one data stream to the other or through filtering. Potential operations include

Filtering and removing some events

Reshaping a stream by removing, renaming, or adding new attributes to a stream

Splitting and combining attributes in a stream

Transforming attributes

For example, from a twitter data stream, we might choose to extract the fields: author, timestamp, location, and then filter them based on the location of the author.

**Pattern 2: Alerts and Thresholds**

This pattern detects a condition and generates alerts based on a condition. (e.g. Alarm on high temperature). These alerts can be based on a simple value or more complex conditions such as rate of increase etc.

For an example, in TFL (Transport for London) Demo video based on transit data from London, we trigger a speed alert when the bus has exceed a given speed limit.

**Pattern 3: Simple Counting and Counting with Windows**

**Pattern 4: Joining Event Streams**

**Pattern 5: Data Correlation, Missing Events, and Erroneous Data**

**Pattern 6: Interacting with Databases**

**Pattern 7: Detecting Temporal Event Sequence Patterns**

**Pattern 8: Tracking ( track something over space or time)**

**Pattern 9: Detecting Trends ( rise, fall, turn, tipple bottom)**

**Pattern 10: Running the same Query in Batch and Realtime Pipelines**

**Pattern 11: Detecting and switching to Detailed Analysis**

**Pattern 12: Using a Machine Learning Model**

**Pattern 13: Online Control**