Master Executive di II Livello

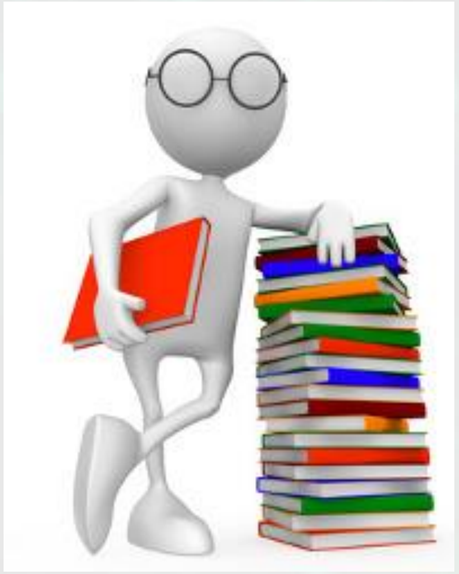
BIG DATA ANALYSIS AND BUSINESS INTELLIGENCE

Stream processing

Vamsi Krishna Varma Gunturi
Data science intern at ISTAT
vamsivarmagunturi@gmail.com

fondazione INUIT TORVERGATA

Topics



- What is stream processing ?
- Why stream processing ?
- Batch processing data flow
- Batch processing vs Stream processing
- Data stream management systems (DSMS)
- DBMS vs DSMS
- Real time data analysis
- Use cases
- Tools for stream processing
- Spark streaming

Stream processing

- Used to query continuous data stream and detect conditions, quickly, within a small time period from the time of receiving the data.
- Stream processing is useful for tasks like fraud detection. If you stream-process transaction data, you can detect anomalies that signal fraud in real time, then stop fraudulent transactions before they are completed.
- Called by many names: real-time analytics, streaming analytics, Complex Event Processing, real-time streaming analytics, and event processing
- It is popularized by Apache Storm, as a “technology like Hadoop but can give you results faster”, after which it was adopted as a Big data technology.

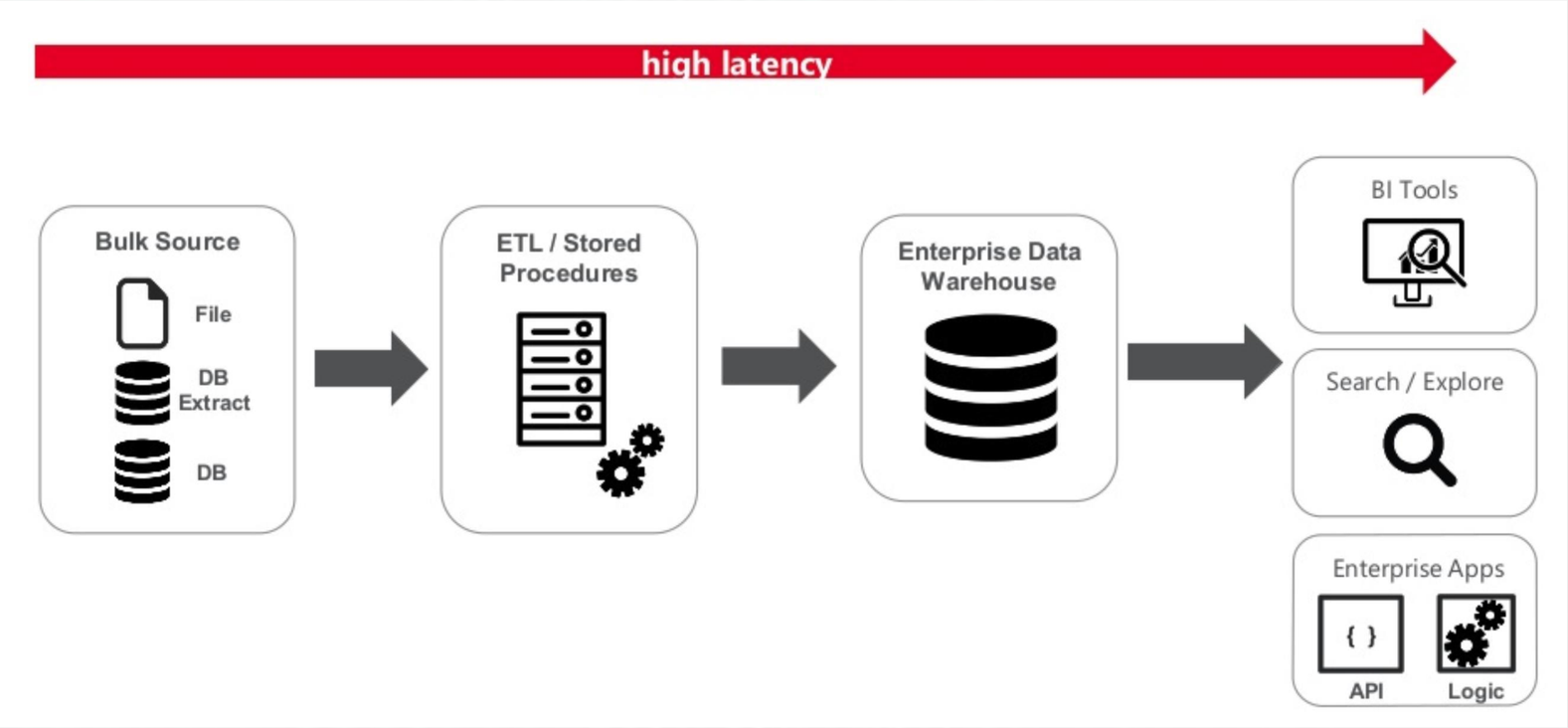
Why Stream processing ?

- Big data established the value of insights derived from processing data. Such insights are not all created equal.
- Some insights are more valuable shortly after it has happened with the value diminishes very fast with time.
- Stream Processing enables such scenarios, providing insights faster, often within milliseconds to seconds from the trigger.
- Streaming handles never ending data streams gracefully and naturally. You can detect patterns, inspect results, look at multiple levels of focus, and also easily look at data from multiple streams simultaneously.
- Stream processing naturally fit with time series data and detecting patterns over time.

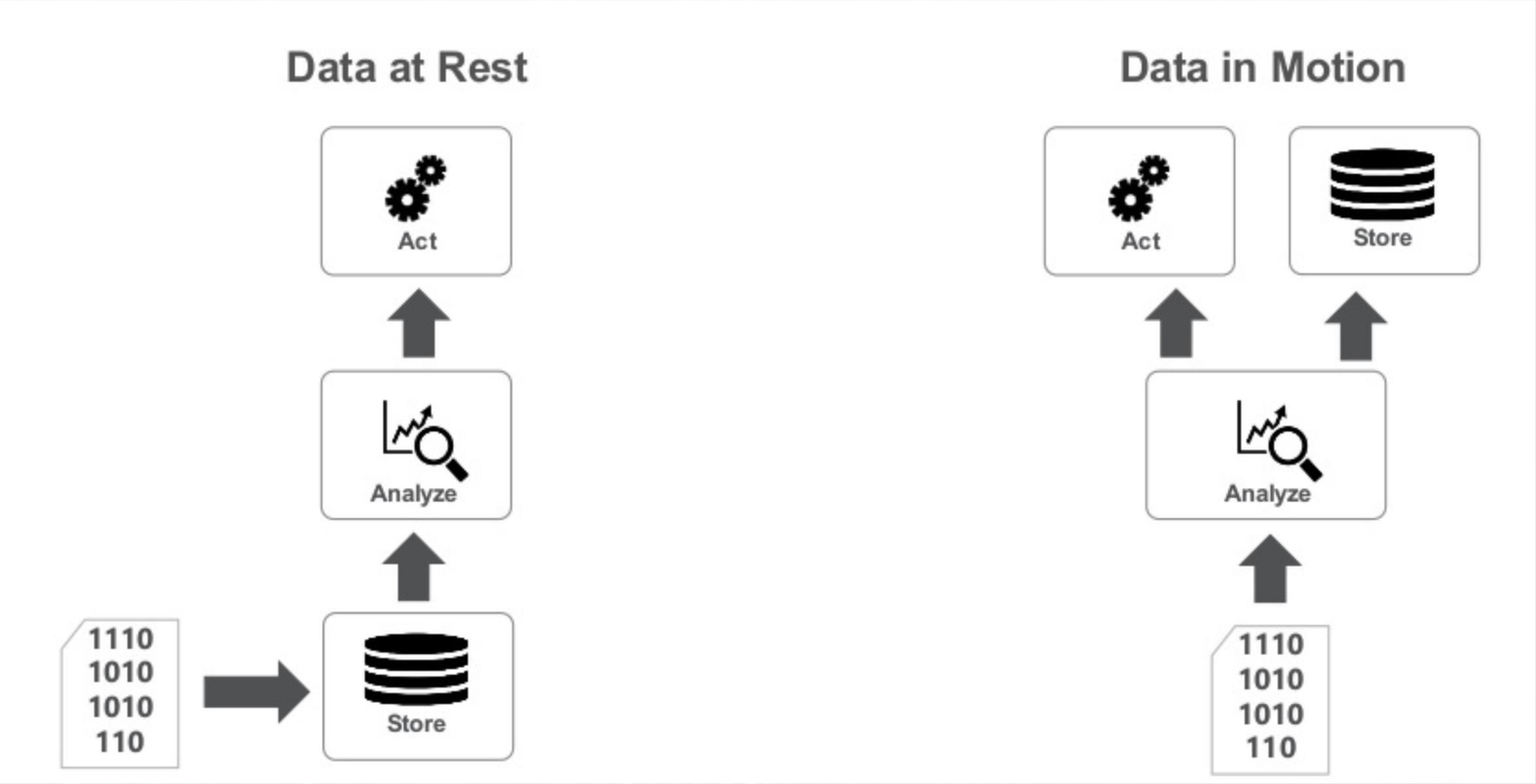
Why Stream processing ?

- The most continuous data series are time series data: traffic sensors, health sensors, transaction logs, activity logs, etc. Almost all IoT data are time series data. Hence, it makes sense to use a programming model that fits naturally.
- Stream processing can work with a lot less hardware than batch processing.
- Stream processing let you handle large fire horse style data and retain only useful bits.
- **Rule of thumb**: if processing can be done with a single pass over the data or has temporal locality (processing tend to access recent data) then it is a good fit for streaming.

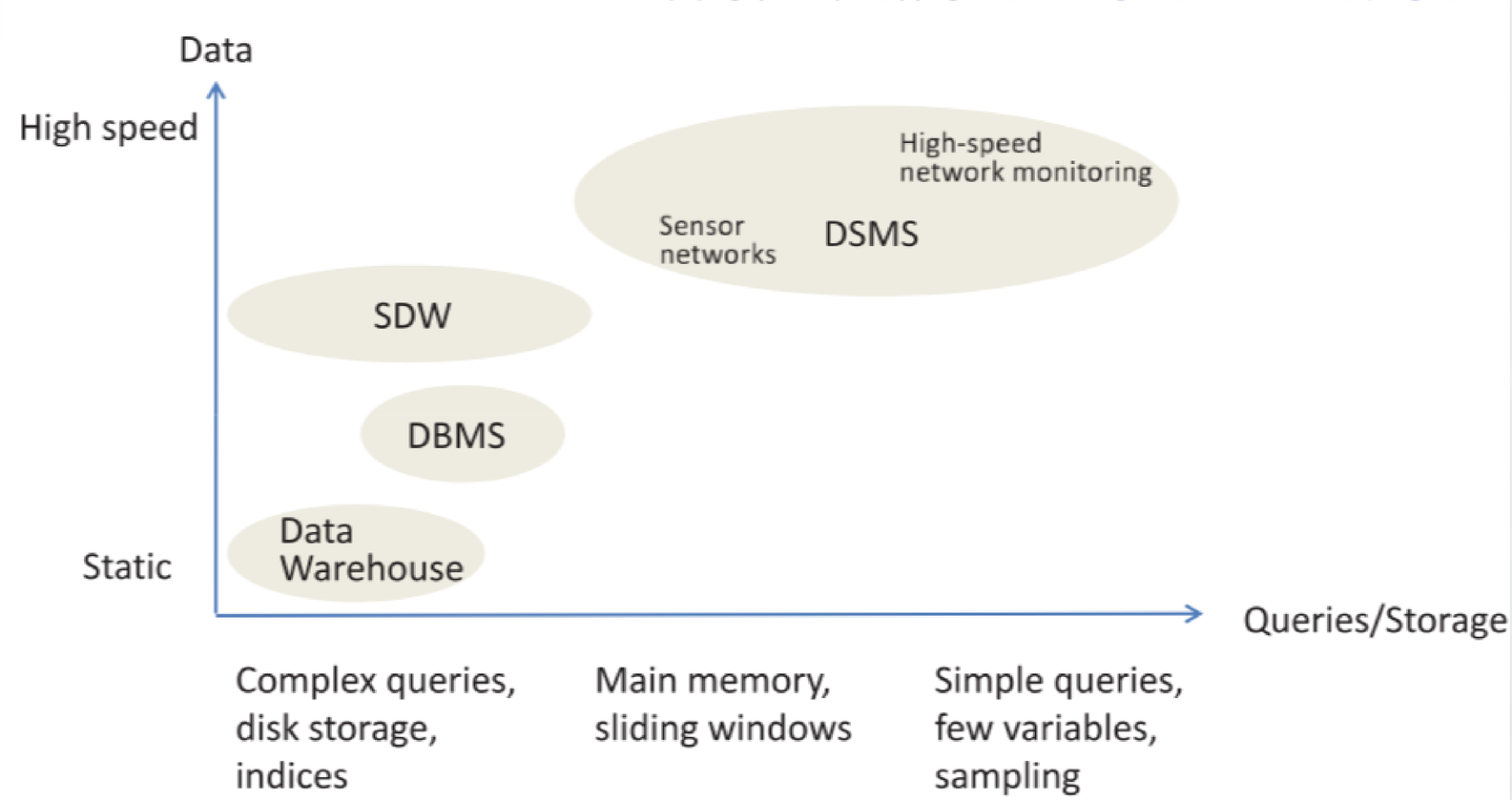
Batch processing workflow



Batch vs Stream processing



Data stream management systems



DBMS vs DSMS

Database management system (DBMS)	Data stream management system (DSMS)
Persistent data (relations)	volatile data streams
Random access	Sequential access
One-time queries	Continuous queries
(theoretically) unlimited secondary storage	limited main memory
Only the current state is relevant	Consideration of the order of the input
relatively low update rate	potentially extremely high update rate
Little or no time requirements	Real-time requirements
Assumes exact data	Assumes outdated/inaccurate data
Plannable query processing	Variable data arrival and data characteristics

Use cases

- Algorithmic Trading, Stock Market Surveillance
- Smart Patient Care
- Monitoring a production line
- Supply chain optimizations
- Intrusion, Surveillance and Fraud Detection (e.g. Uber)
- Most Smart Device Applications: Smart Car, Smart Home
- Traffic Monitoring, Geofencing, Vehicle, and Wildlife tracking
- Sports analytics
- Context-aware promotions and advertising
- Geospatial data processing

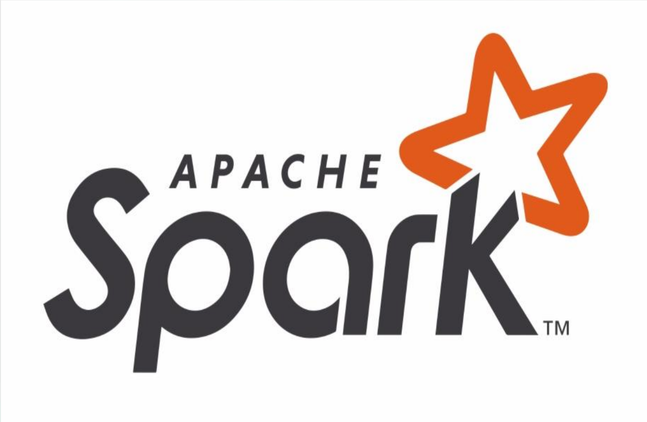
Tools

- Apache Storm
- Apache Kafka
- Apache Spark
- Apache Fink
- Apache Samza
- Streaming SQL
- WSO2 stream processer

Spark streaming

- Analyses continual streams of data
  Example: Processing log data from a website or server
- Data is aggregated and analysed at some interval.
- Can take data fed to some port, Amazon Kinesis, HDFS, Kafka, Flume and others.
- Checkpointing stores state to disk periodically for fault tolerance.
- Python support for Spark streaming is currently incomplete.

Spark streaming



- A “Dstream” object breaks up the stream in to distinct RDD’s
- Example(in Scala):

```
val stream = new StreamingContext(conf, Seconds(1))
val lines = stream.socketTextStream(“localhost”, 8888)
val errors = lines.filter(_.contains(“error”))
errors.print()
```

This listens to log data sent in to port 8888, one second at a time and prints out error lines.

- You need to kick off the job explicitly:

```
stream.start()
stream.awaitTermination()
```

Spark streaming

- Remember your RDD’s only contain one little chunk of incoming data.
- ”Windowed operations” can combine results from multiple batches over some sliding time window. For this we use in-built functions like window(), reduceByWindow(), reduceByKeyAndWindow()

  Example: Top sellers, most frequently viewed pages on your website
- updateStateByKey():
  Lets you maintain a state across many batches as time goes on
  For example: running counts of some event.

Master Executive di II Livello

BIG DATA ANALYSIS AND BUSINESS INTELLIGENCE

Vamsi Krishna Varma Gunturi
Data science intern at ISTAT
vamsivarmagunturi@gmail.com

Grazie

fondazione INUIT TORVERGATA