Guido Schmutz

Working for Trivadis for more than 19 years
Oracle ACE Director for Fusion Middleware and SOA
Co-Author of different books
Consultant, Trainer, Software Architect for Java, SOA & Big Data / Fast Data
Member of Trivadis Architecture Board
Technology Manager @ Trivadis

More than 25 years of software development experience

Contact: guido.schmutz@trivadis.com
Blog: http://guidoschmutz.wordpress.com
Slideshare: http://www.slideshare.net/gschmutz
Twitter: gschmutz

Introduction to Streaming Analytics
Our company.

Trivadis is a market leader in IT consulting, system integration, solution engineering and the provision of IT services focusing on Oracle and Microsoft technologies in Switzerland, Germany, Austria and Denmark. We offer our services in the following strategic business fields:

Trivadis Services takes over the interacting operation of your IT systems.

Introduction to Streaming Analytics
With over 600 specialists and IT experts in your region.

- 14 Trivadis branches and more than 600 employees
- 200 Service Level Agreements
- Over 4,000 training participants
- Research and development budget: CHF 5.0 million
- Financially self-supporting and sustainably profitable
- Experience from more than 1,900 projects per year at over 800 customers
Technology on its own won't help you.
You need to know how to use it properly.
Agenda

1. Introduction / Motivation
2. Concepts (1-11)
3. Stream Processing Platforms
Traditional Data Processing - Challenges

- Introduces too much “decision latency”
- Responses are delivered “after the fact”
- Maximum value of the identified situation is lost
- Decision are made on old and stale data
- “Data a Rest”
Customer 360° View – Flow with Big Data Hub

File Import / SQL Import

Hadoop Cluster
- Distributed Filesystem
- NoSQL
- Parallel Processing
- Machine Learning
- Graph Algorithms
- Natural Language Processing

Data Flow

Event Hub

Data Flow

SQL Export

SQL Export

Enterprise Data Warehouse

BI Tools

Search

Online & Mobile Apps

Introduction to Streaming Analytics
The New Era: Streaming Data Analytics / Fast Data

- Events are analyzed and processed in real-time as they arrive.
- Decisions are timely, contextual and based on fresh data.
- Decision latency is eliminated.
- "Data in motion"
Customer Event Hub – taking Velocity into account

- Billing & Ordering
- CRM / Profile
- Marketing Campaigns
- Location
- Mobile Apps
- Social
- Weather Data
- Click stream
- Call Center
- Sensor Data

Introduction to Streaming Analytics
When to Stream / When not?

- **Constant low**
  - Milliseconds & under

- **Low milliseconds to seconds, delay in case of failures**

- **10s of seconds of more, Re-run in case of failures**

- **Real-Time**
- **Near-Real-Time**
- **Batch**

Introduction to Streaming Analytics
“No free lunch”

- **Constant low**
  - Milliseconds & under

- **Low milliseconds to seconds**, delay in case of failures

- **10s of seconds of more**, Re-run in case of failures

**Real-Time**

**Near-Real-Time**

**Batch**

“**Difficult**” architectures, lower latency

“**Easier architectures**”, higher latency

Introduction to Streaming Analytics
Real Time Analytics Use Cases

- Algorithmic Trading
- Online Fraud Detection
- Geo Fencing
- Proximity/Location Tracking
- Intrusion detection systems
- Traffic Management

- Recommendations
- Churn detection
- Internet of Things (IoT) / Intelligence Sensors
- Social Media/Data Analytics
- Gaming Data Feed
- ...

Introduction to Streaming Analytics
Concepts (1 – 11)
1) Categories of Stream/Event Processing

Simple Event Processing (SEP)

- Single events
- A
- B
- A
- C
- Simple Event Processing (SEP)
- Filter
- Transformation
- Enrichment
- Splitting
- Routing
- A
- A
- Event Consumer
  - Dashboard
  - Business Process
  - Business Service

Event Stream Processing (ESP)

- Event Stream
- A
- A
- A
- A
- A
- A
- Event Stream Processing (ESP)
  - All of SEP +
  - Aggregation
  - High-Volume Streaming
  - Simple Decision Rules
  
- A1
- A1
- A1
- Event Consumer
  - Dashboard
  - Business Process
  - Business Service

Introduction to Streaming Analytics
1) Categories of Stream/Event Processing

Complex Event Processing (CEP)

- Event Cloud
  - A
  - B
  - C
  - D

- Complex Event Processing (CEP)
  - all of SEP + ESP +
    - Pattern Detection
    - Time Window
    - Correlation / Join
    - Location / Motion

- Event Consumer
  - Dashboard
  - Business Process
  - Business Service

Result

Introduction to Streaming Analytics
2) Streaming Model – Continuous Streaming

- Events processed as they arrive
- Low-latency
- Less throughput
- Fault tolerance expensive

Introduction to Streaming Analytics
2) Streaming Model - Micro-Batch

- Splits incoming stream in small batches
- Could provide high(er) throughput
- Fault tolerance easier to achieve
- Higher latency
3) Delivery Guarantees

At most once (fire-and-forget) - means the message is sent, but the sender *doesn't care if it's received or lost*. Imposes no additional overhead to ensure message delivery, such as requiring acknowledgments from consumers. Hence, it is the easiest and most performant behavior to support.

At least once - means that retransmission of a message will occur until an acknowledgment is received. Since a delayed acknowledgment from the receiver could be in flight when the sender retransmits the message, the message may be received one or more times.

Exactly once - ensures that a message is received once and only once, and is never lost and never repeated. The system must implement whatever mechanisms are required to ensure that a message is received and processed just once.
4) API

Compositional / Programmatic
- Highly customizable operator based on basic building blocks
- Manual topology definition and optimization

Declarative
- High-Level, fluent API
- Higher order function as operators (filter, mapWithState …)
- Logical plan optimization

Statistical
- Data Scientist friendly
- Dynamic type

SQL
- Query language allowing to use stream in FROM clause
- Extensions supporting windowing, pattern matching, spatial, … operators
5) Windowing

Due to size and never-ending nature of it, it’s not feasible to keep entire stream of data in memory.

Computations over multiple events can be done using **windows of data**

- **Sliding Window** - uses eviction and trigger policies that are based on time: *window length* and *sliding interval length*

- **Tumbling Window** - eviction policy is always based on the window being full and the trigger policy is based on either the count of items in the window or time
6) Pattern Matching

- Streaming Data often contains interesting patterns that only emerge as new streaming data arrives, e.g.
  - Absence Pattern: event A not followed by event B within time window
  - Sequence Pattern: event A followed by event B followed by event C
  - Increasing Pattern: up trend of a value of a certain attribute
  - Decreasing Pattern: down trend of a value of a certain attribute
  - …

- Pattern operators allow developers to define complex relationships between streaming events
7) Back Pressure

- Backpressure refers to the situation where a system is receiving data at a higher rate than it can process.
- Backpressure, if not dealt with correctly, can lead to exhaustion of resources, or even, in the worst case, data loss.

- A slow consumer should slow down the producer.
Other Concepts

7) Scalability
   • Scale-Up and/or Scale-Out

8) Resource Management
   • Proprietary Server, YARN, Mesos, ….

7) Auto-Scaling:
   • Does platform support scale-up and scale-down without stopping

8) In-Flight Modifications
   • Are modifications on running topology supported without redeployment

9) Contributors
   • How many active contributors

10) Language Options
    • What languages can be used

11) ”Self-Service”
    • Business user friendliness
Stream Processing Platforms
Stream Processing & Analytics

- Complex Event Processing
  - WSO² Complex Event Processor
  - SAP Event Stream Processor
  - TIBCO StreamBase
  - ORACLE FUSION MIDDLEWARE Stream Analytics Platform

- Simple Event Processing
  - Apache Flink
  - Apache SAMZA
  - Apache Storm
  - Apache HERON
  - Apache NIFI
  - Apache QUARKS
  - Apache StreamSets
  - Apache Kafka Streams

- Event Stream Processing
  - Apache Kafka
  - Apache NIFI

- Open Source
- Closed Source

Introduction to Streaming Analytics
Oracle Stream Analytics

What it does
• Compelling, friendly and visually stunning real time streaming analytics user experience for Business users to dynamically create and implement Instant Insight solutions

Key Features
• Analyze simulated or live data feeds to determine event patterns, correlation, aggregation & filtering
• Pattern library for industry specific solutions
• Streams, References, Maps & Explorations

Benefits
• Accelerated delivery time
• Hides all challenges & complexities of underlying real-time event-driven infrastructure

Introduction to Streaming Analytics
Oracle Stream Analytics

- **2007**: BEA Weblogic Event Server
- **2008**: Oracle Complex Event Processing (OCEP)
  - **2012**: Oracle Event Processing (OEP)
    - **2013**: Oracle Stream Explorer (SX)
      - **2015**: Oracle Stream Analytics (OSA)
      - **2016**: Oracle IoT Cloud Service
    - **2013**: Oracle Event Processing for Java Embedded
  - **2016**: Oracle Edge Analytics (OAE)

Introduction to Streaming Analytics
Oracle Stream Analytics

Introduction to Streaming Analytics

**Devices / Gateways**
- Filtering
- Correlation
- Aggregation
- Pattern matching

**EDGE Analytics**
- Macro-event
- High-value
- Actionable
- In-context

**Services**
- Stream Analytics
  - High Volume
  - Continuous Streaming
  - Sub-Millisecond Latency
  - Disparate Sources
  - Time-Window Processing
  - Pattern Matching

**Enterprise**
- High Availability / Scalability
- Coherence Integration
- Geospatial, Geofencing
- Big Data Integration
- Business Event Visualization
- Action!

"Sea of data"
Oracle Stream Analytics

<table>
<thead>
<tr>
<th>Category</th>
<th>CEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Streaming Model</td>
<td>Continuous Streaming</td>
</tr>
<tr>
<td>Delivery Guarantees</td>
<td>At Least Once</td>
</tr>
<tr>
<td>API</td>
<td>Declarative/SQL</td>
</tr>
<tr>
<td>Windowing</td>
<td>Yes</td>
</tr>
<tr>
<td>Pattern detection</td>
<td>Yes</td>
</tr>
<tr>
<td>Back Pressure</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scalability</th>
<th>Scale-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource Management</td>
<td>OEP server, Spark Streaming</td>
</tr>
<tr>
<td>Auto-Scaling</td>
<td>no</td>
</tr>
<tr>
<td>In-Flight modifications</td>
<td>no</td>
</tr>
<tr>
<td>Contributors</td>
<td>n.a.</td>
</tr>
<tr>
<td>Language Options</td>
<td>Java, CQL</td>
</tr>
<tr>
<td>“self-service”</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Apache Spark

Apache Spark is a fast and general engine for large-scale data processing

- The hot trend in Big Data!
- Originally developed 2009 in UC Berkley’s AMPLab
- Based on 2007 Microsoft Dryad paper
- Written in Scala, supports Java, Python, SQL and R
- Can run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk
- One of the largest OSS communities in big data with over 200 contributors in 50+ organizations
- Open Sourced in 2010 – since 2014 part of Apache Software foundation
Apache Spark – Stack

- **Spark SQL** (Batch Processing)
- **Blink DB** (Approximate Querying)
- **Spark Streaming** (Real-Time)
- **MLlib, Spark R** (Machine Learning)
- **GraphX** (Graph Processing)

**Core Runtime**

**Spark Core API and Execution Model**

**Cluster Resource Managers**

- Spark Standalone
- MESOS
- YARN

**Data Stores**

- HDFS
- Elastic Search
- NoSQL
- S3

Introduction to Streaming Analytics
Apache Spark - Batch vs. Real-Time Processing

Introduction to Streaming Analytics
Apache Spark – Resilient Distributed Dataset (RDD)

Input Source
- File
- Database
- Stream
- Collection

```
data = sc.textFile('input_file')
data.count()  # Output: 100
```
Apache Spark - Workflow

Stage 1
- `flatMap()` + `map()`

Transformations (Lazy)

Action (Execute Transformations)

Introduction to Streaming Analytics
Apache Spark Streaming

DStream

X Seconds

Transform

.countByValue()
.reduceByKey()
.join
.map

DStream

Introduction to Streaming Analytics
Apache Spark Streaming

Input Stream

Event DStream

MappedDStream map()

saveAsHadoopFiles()

Actions Trigger Spark Jobs

DStream Transformation Lineage

Transformation

Lineage

Actions

Trigger

Spark Jobs

Adapted from Chris Fregly: http://slidesha.re/11PP7FV

Introduction to Streaming Analytics
Apache Spark Streaming

```scala
class TwitterStreamingHashTagsByInterval extends Serializable {

  def start(auth: Option[Authorization], ssc: StreamingContext, filters: Regex, keyspace: String, table: String): Unit = {
    val transform = (crutf: String) => filters.findAllIn(crutf).flatMap(_.stripPrefix("#"))
    val stream = TwitterUtils.createStream(ssc, auth, Nil, StorageLevel.MEMORY_ONLY_SER_2)

    /** Note that Cassandra is doing the sorting for you here. */
    stream.flatMap(_.getText.toLowerCase.split(""\\s+""))
      .map(transform)
      .countByValueAndWindow(Seconds(5), Seconds(5))
      .transform((rdd, time) => rdd.map { case (term, count) => (term, count, now(time)) })
      .saveToCassandra(keyspace, table, SomeColumns("hashtag", "mentions", "interval"))

    ssc.checkpoint("./checkpoint")
    ssc.start()
    ssc.awaitTermination()
  }

  private def now(time: Time): String =
    new DateTime(time.milliseconds, DateTimeZone.UTC).toString("yyyyMMMdHH:mm:ss.SSS")
```

Introduction to Streaming Analytics
## Spark Streaming

<table>
<thead>
<tr>
<th>Category</th>
<th>ESP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Streaming Model</td>
<td>Continuous Streaming</td>
</tr>
<tr>
<td>Delivery Guarantees</td>
<td>Exactly Once</td>
</tr>
<tr>
<td>API</td>
<td>Declarative/SQL</td>
</tr>
<tr>
<td>Windowing</td>
<td>Yes</td>
</tr>
<tr>
<td>Pattern detection</td>
<td>No</td>
</tr>
<tr>
<td>Back Pressure</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scalability</th>
<th>Scale-out</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource Management</td>
<td>YARN, Mesos, Standalone</td>
</tr>
<tr>
<td>Auto-Scaling</td>
<td>yes</td>
</tr>
<tr>
<td>In-Flight modifications</td>
<td>no</td>
</tr>
<tr>
<td>Contributors</td>
<td>&gt; 280 contributors</td>
</tr>
<tr>
<td>Language Options</td>
<td>Java, Scala, Python</td>
</tr>
<tr>
<td>&quot;self-service&quot;</td>
<td>No</td>
</tr>
</tbody>
</table>

Introduction to Streaming Analytics
Apache Storm

A **platform** for doing analysis on streams of data as they come in, so you can react to data **as it happens**.

- highly distributed real-time computation system
- Provides general primitives to do real-time computation
- To simplify working with queues & workers
- scalable and fault-tolerant

Originated at Backtype, acquired by Twitter in 2011
Open Sourced late 2011
Part of Apache since September 2013
Apache Storm

Tuple
- Immutable Set of Key/value pairs

Stream
- an unbounded sequence of tuples that can be processed in parallel by Storm

Topology
- Wires data and functions via a DAG (directed acyclic graph)
- Executes on many machines similar to a MR job in Hadoop

Spout
- Source of data streams (tuples)
- can be run in “reliable” and “unreliable” mode

Bolt
- Consumes 1+ streams and produces new streams
- Complex operations often require multiple steps and thus multiple bolts

Introduction to Streaming Analytics
Apache Storm

```java
public class SentenceSpout extends BaseRichSpout {
    private SpoutOutputCollector collector;
    private String[] sentences = {"my dog has fleas", "i like cold beverages", "the dog ate my homework", "don't have a cow man", "i don't think i like fleas"};
    private int index = 0;

    public void declareOutputFields(OutputFieldsDeclarer declarer) {
        declarer.declare(new Fields("sentence"));
    }

    public void open(Map conf, TopologyContext context)
            SpoutOutputCollector collector) {
        this.collector = collector;
        index = 0;
    }

    public void nextTuple() {
        if (index >= sentences.length) {
            index = 0;
        }
        String[] sentences = sentences[index++);
        this.collector.emit(new Values(sentences));
        Utils.waitForMillis(100);
    }

    public class WordCountBolt extends BaseRichBolt {
        private HashMap<String, Long> counts = null;

        public void prepare(Map conf, TopologyContext context,
                OutputCollector collector) {
            this.collector = collector;
            this.counts = new HashMap<String, Long>();
        }

        public void execute(Tuple tuple) {
            String word = tuple.getStringByField("word");
            Long count = this.counts.get(word);
            if (count == null) {
                count = 0L;
            }
            count++;
            System.out.println("Count for word ' + word + ' = " + count);
            this.counts.put(word, count);
            this.collector.emit(new Values(word, count));
        }

        public void declareOutputFields(OutputFieldsDeclarer declarer) {
            declarer.declare(new Fields("word", "count"));
        }
    }
}
```

Introduction to Streaming Analytics
## Apache Storm

<table>
<thead>
<tr>
<th>Category</th>
<th>ESP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Streaming Model</td>
<td>Event-Streaming</td>
</tr>
<tr>
<td>Delivery Guarantees</td>
<td>At most once / At least once</td>
</tr>
<tr>
<td>API</td>
<td>Compositional</td>
</tr>
<tr>
<td>Windowing</td>
<td>No</td>
</tr>
<tr>
<td>Pattern detection</td>
<td>No</td>
</tr>
<tr>
<td>Back Pressure</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scalability</th>
<th>Scale-out</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource Management</td>
<td>Storm Cluster, YARN</td>
</tr>
<tr>
<td>Auto-Scaling</td>
<td>no</td>
</tr>
<tr>
<td>In-Flight modifications</td>
<td>partial</td>
</tr>
<tr>
<td>Contributors</td>
<td>&gt; 100 contributors</td>
</tr>
<tr>
<td>Language Options</td>
<td>Java, Clojure, Scala, ...</td>
</tr>
<tr>
<td>&quot;self-service&quot;</td>
<td>No</td>
</tr>
</tbody>
</table>

---

Introduction to Streaming Analytics
## Comparison

<table>
<thead>
<tr>
<th>Category</th>
<th>Oracle Stream Analytics</th>
<th>Spark Streaming</th>
<th>Spark Storm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Streaming Model</td>
<td>Continuous Streaming</td>
<td>Continuous Streaming</td>
<td>Event-Streaming</td>
</tr>
<tr>
<td>Delivery Guarantees</td>
<td>At Least Once</td>
<td>Exactly Once</td>
<td>At most once / At least once</td>
</tr>
<tr>
<td>API</td>
<td>Declarative/SQL</td>
<td>Declarative/SQL</td>
<td>Compositional</td>
</tr>
<tr>
<td>Windowing</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Pattern detection</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Back Pressure</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Scalability</td>
<td>Scale-up</td>
<td>Scale-out</td>
<td>Scale-out</td>
</tr>
<tr>
<td>Resource Management</td>
<td>OEP server, Spark Streaming</td>
<td>YARN, Mesos, Standalone</td>
<td>Storm Cluster, YARN</td>
</tr>
<tr>
<td>Auto-Scaling</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>In-Flight modifications</td>
<td>no</td>
<td>no</td>
<td>partial</td>
</tr>
<tr>
<td>Contributors</td>
<td>n.a.</td>
<td>&gt; 280 contributors</td>
<td>&gt; 100 contributors</td>
</tr>
<tr>
<td>Language Options</td>
<td>Java, CQL</td>
<td>Java, Scala, Python</td>
<td>Java, Clojure, Scala, ...</td>
</tr>
<tr>
<td>&quot;self-service&quot;</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Summary

More and more use cases (such as IoT) make Streaming Analytics necessary

Treat events as events! Infrastructures for handling lots of events are available!

Platforms such as Oracle Stream Analytics enable the business to work directly on streaming data (empower the business analyst) => User Experience of an Excel Sheet on streaming data

Platform such as Apache Strom and Apache Spark Streaming provide a highly-scalable and fault-tolerant infrastructure for streaming analytics => Oracle Stream Analytics can use Spark Streaming as the runtime infrastructure

Platforms such as Kafka provide a high volume event broker infrastructure, a.k.a. Event Hub
Trivadis @ DOAG 2016

- Booth: 3rd Floor – next to the escalator
- Know how, T-Shirts, Contest and Trivadis Power to go
- We look forward to your visit
- Because with Trivadis you always win!