Stream Processing with Apache Apex

Sandeep Deshmukh
Committer @ Apache Apex
sandeep@apache.org

Bhupesh Chawda
Committer @ Apache Apex
Engineer @ DataTorrent Software
bhupesh@apache.org
• Apex Development Model
• Creating an Apex Application
• Developing an Operator
• Connecting operators (DAG - Low level API)
• Running and Monitoring an Apex Application
  • Showcase Apex Features
• High Level (Stream) API
• Q & A
Apache Apex - Overview

- **Apex - Enterprise grade**, unified batch and stream processing engine
- **Highly Performant** - Can reach single digit millisecond end-to-end latency
- **Highly Scalable** - Horizontal scalability (statically as well as dynamically)
- **Fault Tolerant** - Automatically recovers from failures - without manual intervention
- **Stateful** - Guarantees that no state will be lost
- **Easily Operable** - Exposes an easy API for developing *Operators* and *Applications*

More Details: [https://apex.apache.org/docs/apex/](https://apex.apache.org/docs/apex/)
An Apex Application is a DAG
(Directed Acyclic Graph)

- A DAG is composed of vertices (Operators) and edges (Streams).
- A Stream is a sequence of data tuples which connects operators at end-points called Ports
- An Operator takes one or more input streams, performs computations & emits one or more output streams
  - Each operator is USER’s business logic, or built-in operator from our open source library
  - Operator may have multiple instances that run in parallel
Creating the Application

• Create an apex app using maven archetype

  mvn archetype:generate
  -DarchetypeGroupId=org.apache.apex
  -DarchetypeArtifactId=apex-app-archetype -DarchetypeVersion=3.5.0
  -DgroupId=com.example
  -Dpackage=com.example.demo
  -DartifactId=demoApp -Dversion=1.0-SNAPSHOT

More details: https://www.youtube.com/watch?v=z-eeh-tjQrc
Project Structure

- **pom.xml**
  - Project structure and Dependencies
- **Application.java**
  - Defines the Application DAG
- **RandomNumberGenerator.java**
  - Sample operator
- **properties.xml**
  - Configuration for the application
- **ApplicationTest.java**
  - Application test - Runs in local mode
Apex Application Design

Kafka Input → CSV Parser → Simple Dedup → My SQL Output

byte[] → Java Object → Unique Records

Kafka: Topic: test
MySQL: Table: demo
### Apex Malhar Library

**Malhar Operators**

<table>
<thead>
<tr>
<th>Input/Output Operators</th>
<th>Compute Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Systems</td>
<td>Pattern Matching</td>
</tr>
<tr>
<td>RDBMS</td>
<td>Stats &amp; Math</td>
</tr>
<tr>
<td>NoSQL</td>
<td>Machine Learning &amp; Algorithms</td>
</tr>
<tr>
<td>Messaging</td>
<td>Parsers</td>
</tr>
<tr>
<td>In Memory Databases</td>
<td>UI &amp; Charting Operators</td>
</tr>
<tr>
<td>Social Media</td>
<td>Stream Manipulators</td>
</tr>
<tr>
<td>Protocol Read/Write</td>
<td>Query &amp; Scripting</td>
</tr>
</tbody>
</table>

**Also Includes**

- High Level (Stream) API
- Apache Beam Model
- Compatibility - Windowed Operator
- SQL API
- Managed State
  - Spillable Data structures
- Examples

Details: [https://apex.apache.org/docs/malhar/](https://apex.apache.org/docs/malhar/)
## Apex-Malhar

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>apps</td>
<td>Preparing for 3.8.0 development</td>
<td>2 months ago</td>
</tr>
<tr>
<td>benchmark</td>
<td>Add <code>@since</code> tags and update change log for release 3.7.0</td>
<td>a month ago</td>
</tr>
<tr>
<td>contrib</td>
<td>APEXMALHAR-2278.KuduNonTransactionalOutputOperator</td>
<td>9 days ago</td>
</tr>
<tr>
<td>docs</td>
<td>APEXMALHAR-2426: Incorporated Regex Parser documentation review comments</td>
<td>a month ago</td>
</tr>
<tr>
<td>examples</td>
<td>APEXMALHAR-2455 Create example for Kafka 0.9 API exactly-once output</td>
<td>8 days ago</td>
</tr>
<tr>
<td>hive</td>
<td>Preparing for 3.8.0 development</td>
<td>2 months ago</td>
</tr>
<tr>
<td>kafka</td>
<td>Preparing for 3.8.0 development</td>
<td>2 months ago</td>
</tr>
<tr>
<td>library</td>
<td>Merge branch 'APEXMALHAR-2487' of <a href="https://github.com/ilganeli/incubat%E2%80%A6">https://github.com/ilganeli/incubat…</a></td>
<td>2 hours ago</td>
</tr>
<tr>
<td>samples</td>
<td>Preparing for 3.8.0 development</td>
<td>2 months ago</td>
</tr>
<tr>
<td>sql</td>
<td>Preparing for 3.8.0 development</td>
<td>2 months ago</td>
</tr>
<tr>
<td>stream</td>
<td>Preparing for 3.8.0 development</td>
<td>2 months ago</td>
</tr>
</tbody>
</table>

[https://github.com/apache/apex-malhar/](https://github.com/apache/apex-malhar/)
**Apex Application Design**

**Kafka Input** → **CSV Parser** → **Simple Dedup** → **MySQL Output**

- **Kafka**: byte[]
- **CSV Parser**: Java Object
- **Simple Dedup**: Unique Records
- **MySQL Output**

**Apex Application**

- **Kafka**: Topic: test
- **MySQL**: Table: test

* A better version of Dedup operator is available in Malhar. We’ll however, try to develop a simpler variant from scratch for the purpose of this tutorial.
Add Dependencies in pom.xml

- Add dependencies from Apex Malhar
  - malhar-kafka
  - malhar-library
  - malhar-contrib, any external dependencies
  - malhar-stream* (For high level API)
Operator Lifecycle

Flow for Input Adapters

Flow for Generic Operators and Output Adapters

Component::setup(Context context)
Operator::beginWindow(long windowId)
InputOperator::emitTuples()
Operator::endWindow()
Component::teardown()

Component::setup(Context context)
Operator::beginWindow(long windowId)
InputPort::process()
Operator::endWindow()
Component::teardown()
Developing an Operator

• Basic Functionality
  • Java class
    • Extend `BaseOperator`
    • Implement `Operator`
  • Implement `setup()` and `teardown()` methods
  • Implement `beginWindow(..)` and `endWindow()` methods
  • Declare Input and Output ports
  • Implement `process()` method for an Input port

• Advanced Functionality
  • Add checkpoint Listeners
  • Add Activation Listeners
  • Partitioning
  • Idempotency / Exactly Once guarantees

Operator development guide: [https://apex.apache.org/docs/apex/operator_development/](https://apex.apache.org/docs/apex/operator_development/)
private HashSet<Long> uniqueSet = new HashSet<>();

public final transient DefaultOutputPort<Object> uniques = new DefaultOutputPort<>();

public final transient DefaultOutputPort<Object> duplicates = new DefaultOutputPort<>();

public final transient DefaultInputPort<Object> input = new DefaultInputPort<Object>()
{
    @Override
    public void process(T tuple) {
        if (uniqueSet.contains(tuple.getId())) {
            duplicates.emit(tuple);
        } else {
            uniques.emit(tuple);
            uniqueSet.add(tuple.getId());
        }
    }
};
@ApplicationAnnotation(name="Kafka-MySQL-Using-LowLevelAPI")
public class Application implements StreamingApplication
{
    @ SuppressWarnings({"unchecked", "rawtypes"})
    @ Override
    public void populateDAG(DAG dag, Configuration conf)
    {
        KafkaSinglePortInputOperator kafka = dag.addOperator("kafka", KafkaSinglePortInputOperator.class);
        CsvParser parser = dag.addOperator("parser", CsvParser.class);
        Dedup dedup = dag.addOperator("dedup", Dedup.class);
        JdbcPOJOInsertOutputOperator mysql = dag.addOperator("mysql", JdbcPOJOInsertOutputOperator.class);
        dag.addStream("kafka-parser", kafka.outputPort, parser.in);
        dag.addStream("parser-dedup", parser.out, dedup.input);
        dag.addStream("dedup-mysql", dedup.uniques, mysql.input);
    }
}

https://apex.apache.org/docs/apex/application_development/#java-api
Configuring Properties and Attributes

![Configuring Properties and Attributes](https://apex.apache.org/docs/apex/application_packages/#appliance-configuration)
A note on large state management

- Creating an in-memory operator is easy. Ex. Dedup stores all keys in memory.
  - Increases checkpoint state with number of input tuples
  - Eventually will run out of memory
- Solution - Managed State
  - HDFS backed persistent bucketed storage
  - Incrementally checkpointed
  - Supports asynchronous reads and caching
  - Scalable and Fault tolerant
  - Spillable Data Structures
    - Managed state backed data structures
      - Spillable ArrayList
      - Spillable Map
      - etc.

Deduper in Malhar Library: https://apex.apache.org/docs/malhar/operators/deduper/
Partitioning considerations

- **Static vs Dynamic**
  - Stateless vs Stateful operators
  - Distribution of state among the partitions for stateful operators
- **Distribution of tuples to downstream partitions**
  - May change depending on the functionality
  - Can be controlled by the operator developer - Stream Codecs
- **Parallel Partitioning**
  - Isolated partitioned pipelines

https://apex.apache.org/docs/apex/application_development/#partitioning
Fault Tolerance considerations

- Operator state
  - Checkpointed state
  - Decide on what can be omitted - transients
- Idempotency
  - On a processing window
  - Idempotency managers - Maintain per window persistent state in a WAL
- Exactly Once
  - End to end exactly once
  - Idempotency + At-least Once + checkpointed state
  - Support from external systems
    - Transactional - JDBC
    - Redo support - file

https://apex.apache.org/docs/apex/application_development/#fault-tolerance
High Level (Stream) API

- Easier for beginners to start with
- Fluent API
- Smaller learning curve
- Transform methods in one place vs operator library
- DAG API provides flexibility while high-level API provides ease of use

https://www.slideshare.net/ApacheApex/java-high-level-stream-api
@ApplicationAnnotation(name="Kafka-MySQL-Stream-API")
public class MyApplication implements StreamingApplication
{
    @SuppressWarnings("unchecked")
    @Override
    public void populateDAG(DAG dag, Configuration conf)
    {
        CsvParser parser = new CsvParser();
        Dedup dedup = new Dedup<Employee>();
        JdbcPOJOInsertOutputOperator mysql = new JdbcPOJOInsertOutputOperator();

        ApexStream stream = StreamFactory
            .fromKafka("localhost:9093", "test0", Options.name("kafka"))
            .addOperator(parser, parser.in, parser.out, Options.name("parser"))
            .with("schema", "{"separator": ",", "quoteChar": "\\\", "fields": [{{"name":
            .with(Context.PortContext.TUPLE_CLASS, "com.example.kafkaETLApp.Employee")
            .addOperator(dedup, dedup.input, dedup.uniques, Options.name("dedup"))
            .with(Context.PortContext.TUPLE_CLASS, "com.example.kafkaETLApp.Employee")
            .endWith(mysql, mysql.input, Options.name("mysql"))
            .with("databaseDriver", "com.mysql.jdbc.Driver")
            .with("databaseUrl", "jdbc:mysql://localhost:3306/test")
            .with("username", "root")
            .with("password", "root")
            .with("batchSize", "5")
            .with("tablename", "emp")
            .with(Context.PortContext.TUPLE_CLASS, "com.example.kafkaETLApp.Employee")
            .endWith())
            .populateDag(dag);
    }
}
High Level API - Current Status

- Method chain for readability
- Stateless transform(map, flatmap, filter)
- Some input and output are available (file, console, Kafka)
- Some interoperability (addOperator, getDag, set property/attributes etc)
- Local mode and distributed mode
- Anonymous function class support
- Windowed Stream
  - A few window transforms available (count, reduce, etc)
  - Window types (fix window, sliding window, session window)
  - Trigger types supported (early trigger, late trigger, at watermark)
  - Accumulation modes supported (accumulate, discard, accumulation_retraction)
  - In memory window state (checkpointed)
Resources

- Apache Apex Subscribe - [http://apex.apache.org/community.html](http://apex.apache.org/community.html)
- Presentations
  - Apache Apex - [http://www.slideshare.net/ApacheApex/presentations](http://www.slideshare.net/ApacheApex/presentations)
  - DataTorrent - [http://slideshare.net/DataTorrent/presentations](http://slideshare.net/DataTorrent/presentations)
- Download
  - DataTorrent RTS - [https://datatorrent.com/download/](https://datatorrent.com/download/)
- Twitter
  - @ApacheApex [https://twitter.com/apacheapex](https://twitter.com/apacheapex)
  - @DataTorrent [https://twitter.com/datatorrent](https://twitter.com/datatorrent)
- Webinars - [https://www.datatorrent.com/webinars/](https://www.datatorrent.com/webinars/)
- Videos and Tutorials - [https://www.youtube.com/user/DataTorrent](https://www.youtube.com/user/DataTorrent)
Thank You!

Questions?