**Benchmarking Beam Runners**

This repository is only for research and development. The Beam application in this branch is agnostic to the type of the “Runner” engine executing the application code. The target runners are Flink & Spark. The current code is executing FlinkRunner and the Beam app configuration has a hard coded FlinkRunner value set by design for testing purposes. The future enhancements to make this code 100% runner agnostic will receive the type of the underlying Runner i.e. FlinkRunner or SparkRunner argument from the execution command line. For an understanding of the Flink & Apark runners:

Please see the [Flink Runner in the Beam repository](https://github.com/apache/incubator-beam/tree/master/runners/flink) &   
 [Spark Runner in the Beam repository](https://github.com/apache/incubator-beam/tree/master/runners/spark).

**Spark & Flink Dataflows**

Spark & Flink Dataflows are Runners for Google Dataflow (aka Apache Beam™) which enables you to run Dataflow programs with Apache Spark & Apache Flink™. It integrates seamlessly with the Dataflow API, allowing you to execute Dataflow programs in streaming or batch mode.  
The current Beam application in this repository executed a streaming mode of the incoming big data.

**Beam Streaming Application**

In order to stress the underlying Beam streaming runner engines, the Linear Road problem has been implemented to execute at the runtime. Please refer to the following link to get familiar with Linear Road problem.  
[Linear Road Problem](http://www.isys.ucl.ac.be/vldb04/eProceedings/contents/pdf/RS12P1.PDF)

**Fault-Tolerance**

The current program's state is persisted by Apache Flink™ Cluster. You may re-run and resume your program upon failure or if you decide to continue computation at a later time.

**Sources and Sinks**

The Data Source for this Beam Streaming application is provided by Apache Kafka. A flat file containing Linear Road records i.e. tuples sends the LR records to Kafka Zookeeper at certain port & corresponds to certain topic as a part of Kafka’s Pub-Sub data distribution model. The Beam application subscribes to Kafka Zookeeper port & subsequently receives LR streaming data. It then executes Beam APIs utilizing the underlying Beam runner i.e. FlinkRunner. Upon creating the expected output, it Sinks the output data in local flat files.

**Kafka & Redis integration**

To execute a Dataflow program in streaming mode the streaming mode in the PipelineOptions is enabled:

options.setStreaming(true);

To integrate the Beam app running in Flink Cluster with Kafka, the Beam KafkaIO() is invoked. The received data is stored in a Beam Pcollection object. To save the output data, Java PrintWriter is utilized to write into a flat file. Open Source Redis is integrated with the Beam app to contain the runtime required in-memory information. Redis database gets purged as per each fresh execution of the Beam app.

**Batch**

**TBD**

**Getting Started**

**Install Flink-Dataflow**

TBD

**Running the LR Beam application on a Flink cluster (TBD)**

You can run your Dataflow program on an Apache Flink cluster. Please start off by creating a new Maven project.

mvn archetype:generate -DgroupId=com.mycompany.dataflow -DartifactId=dataflow-test \

-DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

The contents of the root pom.xml should be slightly changed aftewards (explanation below):

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.mycompany.dataflow</groupId>

<artifactId>dataflow-test</artifactId>

<version>1.0</version>

<dependencies>

<dependency>

<groupId>com.dataartisans</groupId>

<artifactId>flink-dataflow\_2.10</artifactId>

<version>0.3.0</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-shade-plugin</artifactId>

<version>2.4.1</version>

<executions>

<execution>

<phase>package</phase>

<goals>

<goal>shade</goal>

</goals>

<configuration>

<transformers>

<transformer implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">

<mainClass>com.dataartisans.flink.dataflow.examples.WordCount</mainClass>

</transformer>

</transformers>

<artifactSet>

<excludes>

<exclude>org.apache.flink:\*</exclude>

</excludes>

</artifactSet>

</configuration>

</execution>

</executions>

</plugin>

</plugins>

</build>

</project>

The following changes have been made:

1. The Flink Dataflow Runner was added as a dependency.
2. The Maven Shade plugin was added to build a fat jar.

A fat jar is necessary if you want to submit your Dataflow code to a Flink cluster. The fat jar includes your program code but also Dataflow code which is necessary during runtime. Note that this step is necessary because the Dataflow Runner is not part of Flink.

You can then build the jar using mvn clean package. Please submit the fat jar in the target folder to the Flink cluster using the command-line utility like so:

./bin/flink run /path/to/fat.jar

**More**

For more information, please visit the [Apache Flink Website](http://flink.apache.org/) or contact the [Mailinglists](http://flink.apache.org/community.html#mailing-lists).

**Disclaimer**

Apache®, Apache Flink™, Flink™, and the Apache feather logo are trademarks of [The Apache Software Foundation](http://apache.org/).