Word Count

Demo Application Guide

Version 1.0

March, 2014

[Introduction 3](#_Toc381623163)

[Pre-Requisites 3](#_Toc381623164)

[Hadoop Services 3](#_Toc381623165)

[dtgateway Service 3](#_Toc381623166)

[Assumptions 4](#_Toc381623167)

[DataTorrent Install Location 4](#_Toc381623168)

[Location of Packaged Demo Applications 4](#_Toc381623169)

[Launching the Demo Applications (demos.jar) using dtcli 4](#_Toc381623170)

[Terms Used 4](#_Toc381623171)

[Word-Count Demo Application 5](#_Toc381623172)

[Word-Count Application Functionality 5](#_Toc381623173)

[Application Directed Acyclic Graph (DAG) 5](#_Toc381623174)

[Operators 5](#_Toc381623175)

[Streams 6](#_Toc381623176)

[Application Source Code 6](#_Toc381623177)

[How to launch the application 6](#_Toc381623178)

[Monitoring the Word-Count Application 6](#_Toc381623179)

[Application Details View 7](#_Toc381623180)

[Viewing Word Count Application Results 8](#_Toc381623181)

[Looking at the Physical Dashboard 8](#_Toc381623182)

[Looking at Container details 9](#_Toc381623183)

[Looking at the generated Log File for displayed results 9](#_Toc381623184)

[Glossary of Terms used 11](#_Toc381623185)

# Introduction

Word-Count application is a simple application that demonstrates DataTorrent Platform’s one of the core features – the streaming window.

Streaming window is one of the basic building blocks provided by the DataTorrent Platform. A streaming window is a slice of time that encapsulates a set of tuples. The collection of these tuples constitutes a window data set, which is also called as an atomic micro-batch.

It allows the application developers to carry out real-time stream computations for a specific time period. The default time period of a streaming window on DataTorrent platform is 500ms.

This application counts the number of unique words in stream of input words for every streaming window. The application then outputs the count to output console.

# Pre-Requisites

The document assumes that following Hadoop and DataTorrent services are running before using this document,

## Hadoop Services

In order to be able to launch Demo Applications on the DataTorrent platform, Hadoop 2.2 (Yarn) grid needs to be running. The Hadoop install can be from any of the distributors or from Apache (<http://hadoop.apache.org/>) as long as it is Hadoop 2.2.

## dtgateway Service

Please make sure that “dtgateway” service is running. This can be checked by executing following command on the command shell,

$ sudo service dtgateway status

# Assumptions

This document assumes following about the location where the Demo applications are placed.

## Location of Packaged Demo Applications

DataTorrent platform ships with a number of Demo applications. You will find a Jar file (demos.jar) containing Demo applications under DataTorrent installation directory. The Jar file demos.jar contains multiple pre-built demo applications that user can launch on a Hadoop cluster. The DataTorrent command line interface (dtcli) also present in the DataTorrent installation folder can be used to launch these Demo applications on a Hadoop cluster. Do read the guides provided with each demo to better understand the features being demonstrated.

## Launching the Demo Applications (demos.jar) using “dtcli”

Please refer to Getting-Started-Guide to find out more about how to use “dtcli” and also how to launch the demo applications (demos.jar) that are shipped with DataTorrent platform.

# Terms Used

To find out more about the terms used in this document, refer to section [Glossary](#_Glossary_of_Terms) in this document.

# Word-Count Demo Application

DataTorrent Platform ships with Word Count Application. Following sections talk about following,

1. [Word-Count Application Functionality](#_Word-Count_Application_Functionalit)
2. [How to launch the application](#_How_to_launch)
3. [How to monitor the application using DataTorrent Gateway Console](#_Monitoring_the_Word-Count)
4. [How to view the Results and Logs](#_Viewing_Word_Count)

## Word-Count Application Functionality

This application counts the number of unique words in stream of input words. Application reads a sample text file repeatedly so as to simulate the stream of input words. On reading the words from input stream, it counts unique words for each streaming window. The streaming window duration is 500ms. For example, assume that in a streaming window, following stream of words is received as –

{“we”, “are”, “not”, “going”, “to”, “have”, “a”, “trade”, “war”, “but”, “we,” “can”, “not”, “have”, “a”, “trade”, “surrender”, “either”},

then at the end of streaming window, application generates the unique word counts as –

{we=2, are=1, not=2, going=1, to=1, have=2, a=2, trade=2, war=1, but=1, can=1, surrender=1, either=1}.

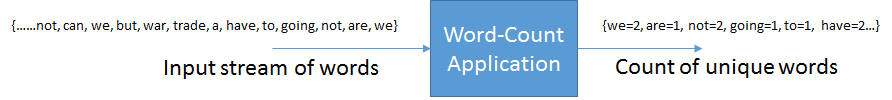


Figure 1 Application Data Flow

Application can also be changed to accept stream of words from any external entity instead of reading it from a file. For example, tweets from Twitter Firehose or Streaming API can be passed to it and get the count of unique words across all the tweets.

### Application Directed Acyclic Graph (DAG)

Word-Count application is represented as following DAG,

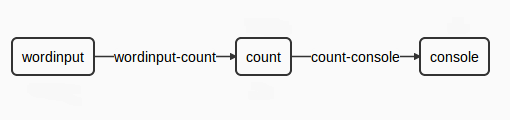


Figure 2 Directed Acyclic Graph

Following sections describe the Operators and Streams that are used to perform the desired processing.

### Operators

As shown in the above DAG, there are three operators that get used. This section explains more about these Operators,

|  |  |
| --- | --- |
| **Operator** | **Description** |
| wordInput | Operator that reads the input text file and generates a stream of words. |
| count | Operator that counts the unique words from the incoming stream of words. |
| console | Displays the count of unique words on the stdout console. |

### Streams

|  |  |
| --- | --- |
| **Stream** | **Description** |
| wordinput-count | Stream of words as input for the count operator. |
| count-console | Stream of counts of unique words as input for the console operator. |

### Application Source Code

Source code for the operators used in the application can be found at,

|  |  |  |
| --- | --- | --- |
| **Operator** | **Source Code** | **Library** |
| wordInput | [WordCountInputOperator.java](https://github.com/DataTorrent/Malhar/blob/master/demos/src/main/java/com/datatorrent/demos/wordcount/WordCountInputOperator.java) | Custom |
| count | [UniqueCounter.java](https://github.com/DataTorrent/Malhar/blob/master/library/src/main/java/com/datatorrent/lib/algo/UniqueCounter.java) | Malhar |
| console | [ConsoleOutputOperator.java](https://github.com/DataTorrent/Malhar/blob/master/library/src/main/java/com/datatorrent/lib/io/ConsoleOutputOperator.java) | Malhar |

The application source code can be viewed at,

* [Application.java](https://github.com/DataTorrent/Malhar/blob/master/demos/src/main/java/com/datatorrent/demos/wordcount/Application.java)

## How to launch the application

The demo applications jar contains the application. On launching the jar from “dtcli” the list of applications packaged in the jar appears as shown in the screenshot below. The entry that is shown for each application is the path of the application class unless an alias was specified for the class in “dt-site.xml” in which case the alias is shown. An integer Id is also shown for each application. The Twitter Application Id appears to be Id “8” in this case as can be seen in screenshot corresponding to the application class. Please specify this Id “8” on the “dtcli” ” command shell to launch the application, as can be seen towards the end in the screenshot.

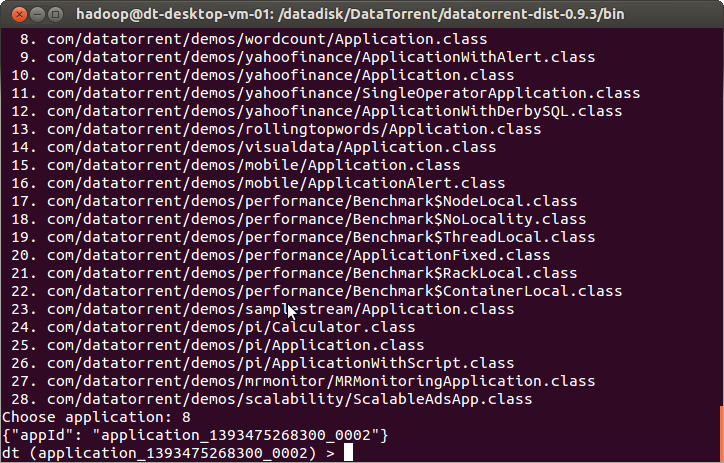


Figure 3 Launch Word-Count Application with AppId - 8

The launched application can be monitored using the DataTorrent Gateway Console as explained below.

## Monitoring the Word-Count Application

Following screenshot displays the launched Word-Count Application in DataTorrent Gateway Console,

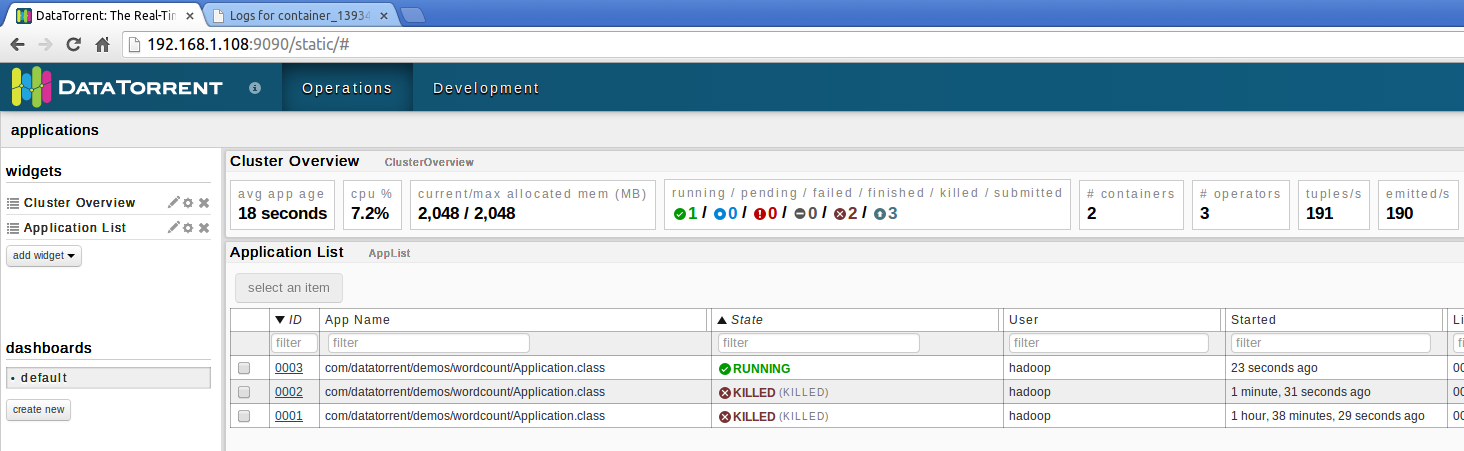


Figure 4 Word-Count Application

The application will be displayed with state as “Running”. The above screen also displays information such as,

1. Time when application was started
2. Memory used by application

To get to further details regarding the application, click on the “Application Id” link under the “Id” column.

### Application Details View

Following screenshot shows details about Word-Count Application. It displays “*logical*” tab with details such,

1. Application Id and Version
2. Application State (Running/Finished etc.)
3. Current Window Id
4. Recovery Window Id
5. Average number of Tuples Processed per second
6. Average number of Tuples Emitted per second
7. Logical Directed Acyclic Graph (DAG)
8. Logical Operators etc.

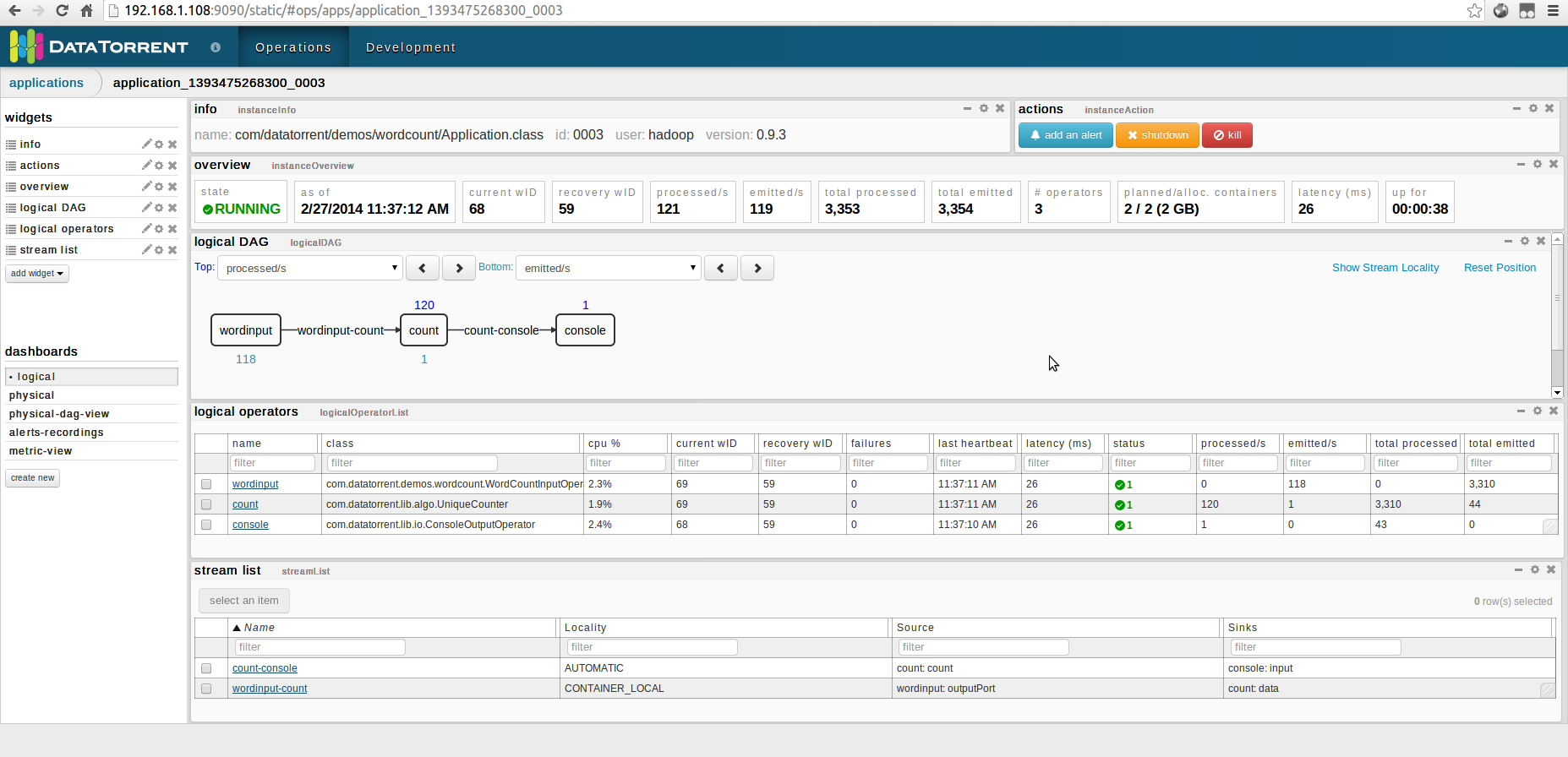


Figure 5 Word-Count Application Details View

## Viewing Word Count Application Results

As the Word-Count application is running, it generates the counts for unique words. It prints these counts on the stdout console through “console” Operator. These results get redirected to log files that are generated for each running application. Following sections explain how to view these generated log files.

### Looking at the Physical Dashboard

Click on the “physical” dashboard, under “dashboards” pane in the left hand bottom pane as seen in the Figure 5 above. This will display the screen that contains a list of “physical operators” (Figure 6 below). This list displays all the operator instances along with Container Ids and the Hadoop nodes where the container process is running.

This list contains the “console” operator which is responsible for displaying the application results. The “console” operator prints the results into the log file that gets generated on the same physical Hadoop node where the “container” process is running.

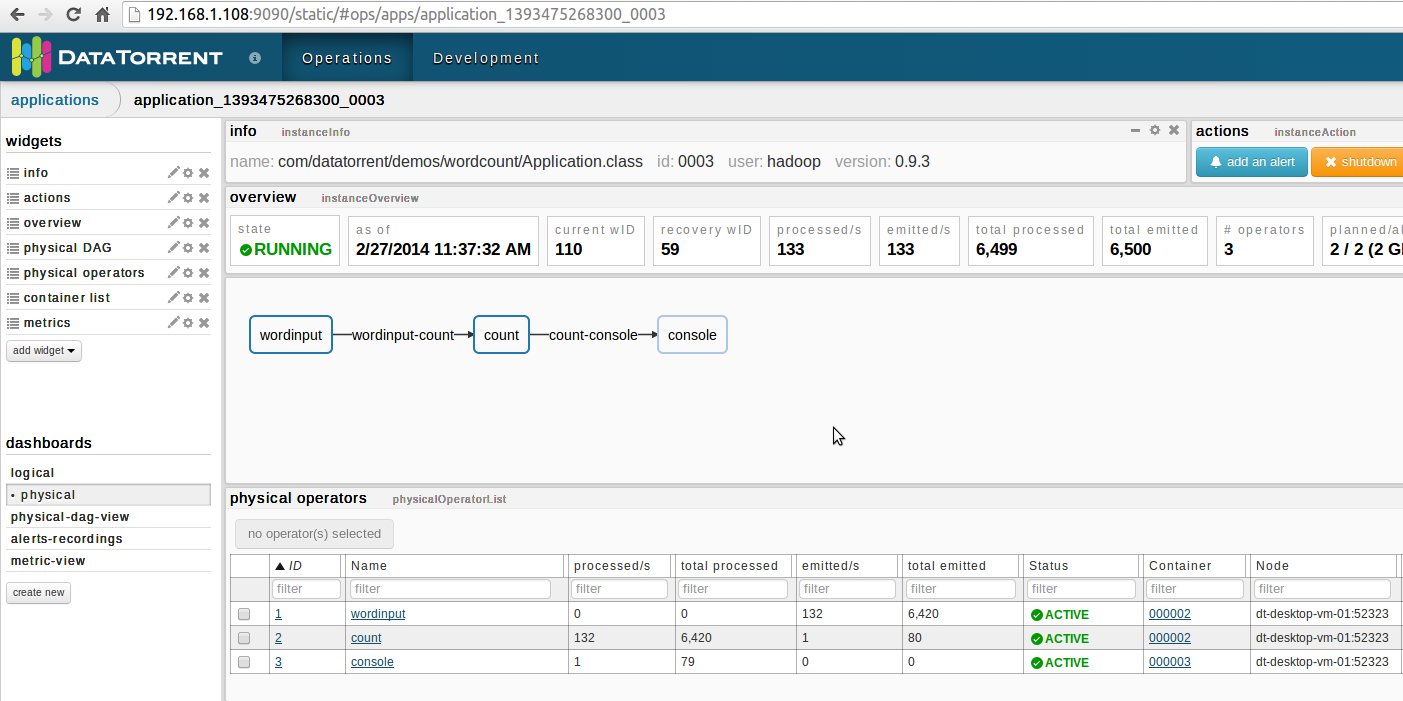


Figure 6 Physical Dashboard - Physical Operators List

### Looking at Container details

In order to get to the log files, we need to get to “container” details by clicking on the container Id (in this case “000003”) for the “console” operator in the “physical operators” list in above Figure 6. That will bring up the following screen as shown in Figure 7 below,

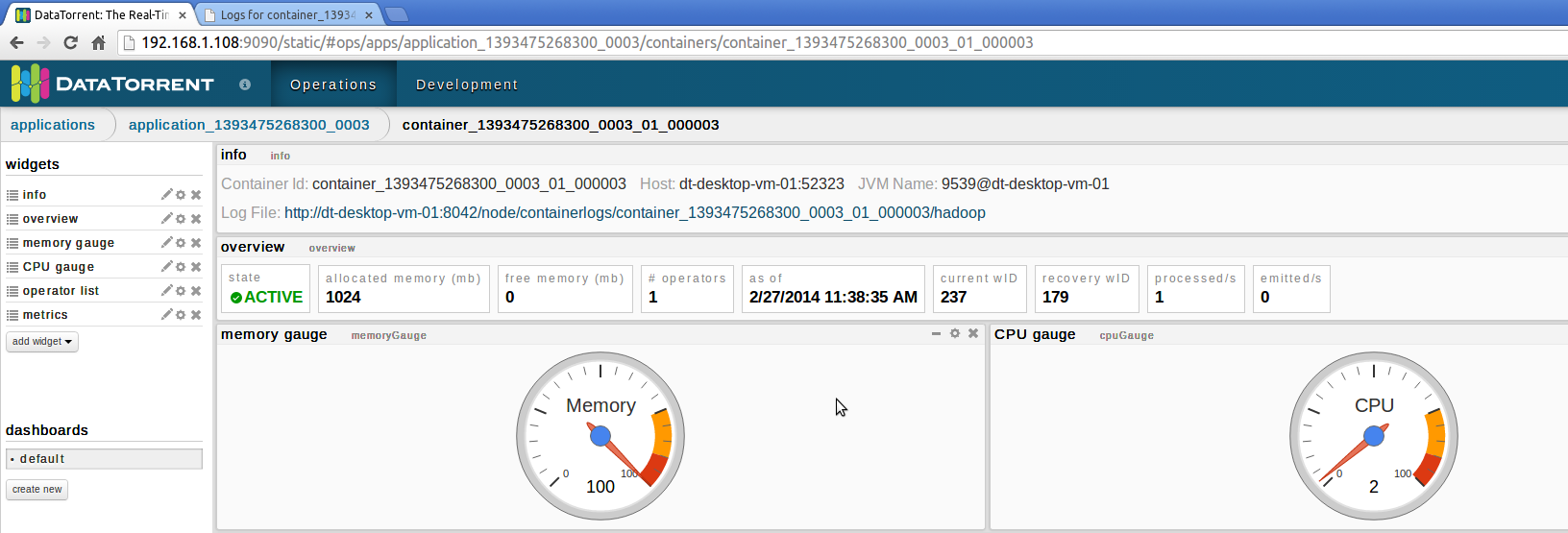


Figure 7 Container Overview

### Looking at the generated Log File for displayed results

Click on the “Log File” link in the “info” widget on the screen displayed in Figure 7 above. That would display the screen that displays names of three log files,

1. DT.log
2. stderr
3. stdout



Figure 8 Container Log Files

Click on the “stdout” link as seen in the above Figure 8. That should bring a screen that looks like following screenshot,



Figure 9 Container stdout log file

You can click on the “here” link to view the entire log file and see the generated results.

# 

# Glossary of Terms used

###### Apache Hadoop

Apache Hadoop is the de-facto distributed operating system, details of which can be found on the web at - <http://apache.hadoop.org/>

###### Hadoop – YARN

Apache Hadoop YARN (Yet Another Resource Negotiator) is a cluster management technology, details of which can be found at - <http://hadoop.apache.org/docs/current2/hadoop-yarn/hadoop-yarn-site/YARN.html>

###### Resource Manager

This is Hadoop-YARN component that allocates and arbitrates the resources such as CPU, Memory and Network.

###### Application

The real-time stream processing application running on DataTorrent platform.

###### Streaming Window

A streaming window is a slice of time that encapsulates a set of tuples. The collection of these tuples constitutes a window data set, which is also called as an atomic micro-batch.

###### Container

A physical resource as allocated by Hadoop-YARN’s Resource Manager.

###### Directed Acyclic Graph (DAG)

It is the logical representation of real-time stream processing application. The computational units within DAG are called as operators and the data-flow edges are called as data streams.

###### Operator

An entity that holds a computational logic to process the data tuples. It is part of a real-time stream processing application. The Operator computational logic gets executed inside a container.

###### Port

Each operator can have ports on which it can receive input data tuples and also output processed data tuples.

###### Stream

A stream consists of data tuples that flow from one port of an operator to another.

###### Logical DAG

Same as “Directed Acyclic Graph” above.

###### Physical DAG

It’s the physical representation of the Logical DAG of the application and is a blueprint of how the application will run on the DataTorrent Cluster’s physical container and nodes.

###### Physical Operator

It’s the physical representation of the operator and provides information such as the name of container and the Hadoop node where operator instance is running.

###### Container

Container represents an allocated resource in the Hadoop cluster.

###### Data Tuples Processed

Number of data objects processed by a real-time stream processing applications

###### Data Tuples Emitted

Number of data objects emitted after the processing is complete by a real-time stream processing applications

###### Current Window Id

The unique identifier for a real-time streaming computation time period

###### Recovery Window Id

The unique identifier for a real-time streaming computation time period up to which the checkpointed state is held into HDFS