A Report on Apache Apex

SRIKANTH RAMANAM[1]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: srikrama@iu.edu*

April 12, 2017

Apache Apex is a Hadoop YARN native big data processing platform with both stream and batch processing capabilities. This paper explores the architecture, functioning and competition of Apache Apex.

Keywords: Stream, Processing, YARN, Apache, Apex, Malhar, I524

https://github.com/cloudmesh/sp17-i524/raw/master/paper2/S17-IR-2028/report.pdf

1. INTRODUCTION

Apex is an enterprise-grade stream and batch processing platform for the Apache Hadoop ecosystem [1]. It was initially developed by DataTorrent as the core engine for RTS,a data processing and analytics platform. Apex was submitted to Apache incubator in 2015 [2]. Later several enterprises like CapitalOne, DirecTV, General Electric, Apple and Silver Spring Networks joined its open source community. Apache Apex was first released in 2016.

Apache Apex is built over YARN and is compatible with existing Hadoop platforms, allowing users to leverage their previous Hadoop investments and applications [3]. According to Apache Apex website [4], " it processes big data in-motion in a way that is highly scalable, highly performant, fault tolerant, stateful, secure, distributed, and easily operable". Apex automatically handles operational aspects like state management, fault tolerance, scalabilty etc [5]. It also provides a simple API that supports Java, facilitating easy development and widespread adoption [5]. It also has a REST API facilitating compatibility with popular web technologies. Apache Apex also has a metrics API that allows users to monitor various aspects of operators in real time.

2. COMPONENTS

Apache Apex has two main components. They are Apex Core and Apex Malhar [6].

2.1. Apex Core

Apex Core is the framework for building distributed stream processing and analytics applications on Hadoop. It also enables building of unified batch and stream processing applications.

2.2. Apex Malhar

Malhar provides a library of operators that perform widely used functionality. These reusable blocks reduce the amount of coding required to build applications and enable users to speed up application development. Operators offered by are mainly of two types

2.2.1. Input/Output Operators

Input/Output operators: These operators offer connectivity with a variety of existing data sources.

2.2.2. Compute Operators

Compute Operators: These operators offer functionality of Machine Learning, Stats and Math, Pattern Marching, Query and Scripting, Stream manipulators, Parsers and UI & Charting.

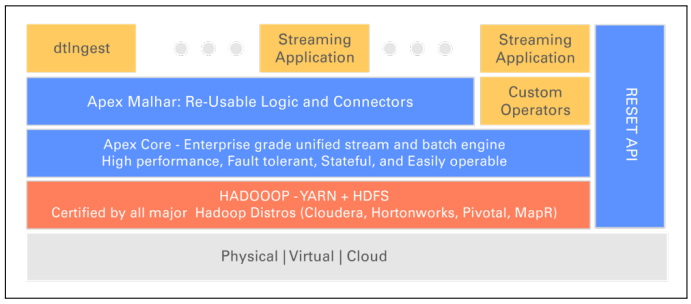

Fig. 1. Apache Apex Components [3]

3. ARCHITECTURE

Operators are the basic blocks of Apex applications. A streaming application is built using in-built or custom operators are connected to form a DAG (Directed Acyclic Graph) using streams.

4. APPLICATION DEVELOPMENT

Apex applications can be written in Java using any IDE supporting Java like Eclipse. Other prerequisites include Apache Maven

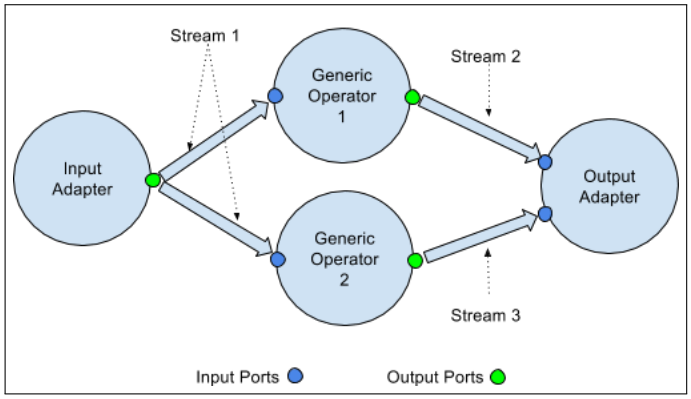

Fig. 2. Apex Application DAG [7]

3.0., Apache Apex, Apache Malhar [7].

4.1. Operators

Operators are independent units of logical operations that either contribute to a part of or a whole business use case. An operator has an input port to receive data tuples and an output port to send data tuples to another operator or external system [8].

4.1.1. Types of Operators [8]

- Input Adapter: An operator at the beginning of the DAG to receive data from an external system.
- Generic Operator: Accepts tuples from previous operator in DAG and does some processing task and outputs the processed data to another operator.
- An operator at the end of a DAG and outputs the data tuples to an external system.

4.1.2. Operator API [8]

- setup() initializes the operator.
- process() performs the core processing operations on data tuples and gets triggered when tuples are received.
- beginWindow() and endWindow() are used for pre and post processing steps.
- teardown() shuts down the operator and releases the resources held by the operator.

4.2. Directed Acyclic Graph

A Directed Acyclic Graph(DAG), is constructed to accomplish a business task using several operators connected through streams [7]. A stream is a sequence of data tuples. To construct a DAG, operators are added using dag.addOperator(args) while streams are added using dag.addStream(args). Other configurations related to YARN can also be added to DAG [9].

4.3. Package

Apex applications are assembled and shared using Apache Apex Packages, which are zip files with all necessary files to launch those applications. Apache Apex Packages are created using Maven. First a Maven project is created with path to the application code. Then a mvn package command creates an application package in the target package. Zip structure of a mvn package consists of app with jar files of the DAG code, lib with jar files of dependencies, conf with preset configurations, META-INF consisting of meta information in files and resources for other files [10].

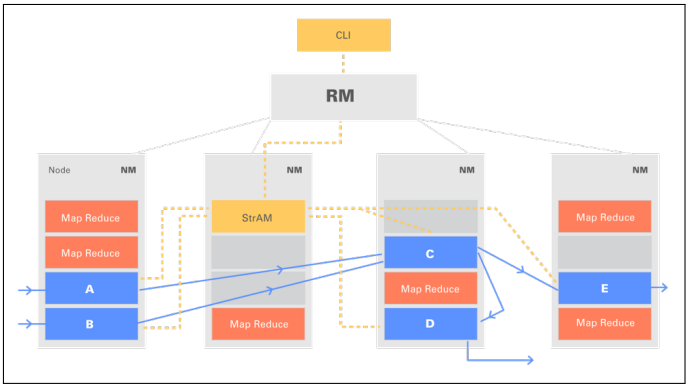

Fig. 3. Apache Apex Application Example [3]

5. FUNCTIONING

Apache Apex applications are built as DAGs consisting of operators and streams. These applications are packaged, shared and deployed over Hadoop clusters.

5.1. Fault tolerance and Recovery

The states of operators and application master are checkpointed regularly to a persistent store like HDFS [11]. Failed operators are automatically detected through continuous monitoring [11]. When a failure occurs, the checkpointed states of operators are used to revive the application [11]. Data can be made to replay from the checkpointed state of the operator after recovery preventing data loss [11].

5.2. Compatibility

Apache Apex is also compatible with several popular data file systems, message systems and database systems through connectors provided by Malhar. They are shown in the below figure.

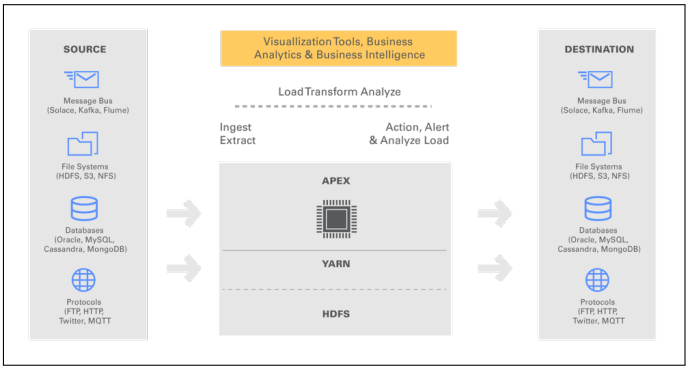

Fig. 4. Apache Apex Interoperability[3]

6. USER INTERFACE

A command line interface called Apex CLI is available for Apache Apex [12]. It can be launched with command apex. help

command is available for all commands to obtain information and syntax.

Apache Apex's REST, Java API's can be integrated with commonly used existing web technologies to create web interfaces for visual analytics. Data Torrent also offers RTS platform, built on Apex, which provides visualization with several real-time dashboards that help monitor streaming applications [11].

7. LICENSE AND PRICING

Apache Apex is a free and open source software. It is licensed under Apache License 2.0 [4].

8. COMPETITION

Some of the competitors of Apache Apex for stream processing and analytics are [13]:

8.1. Apache Spark

This is a large-scale data processing engine that also offers stream processing [14]. But this is not a pure streaming engine as it accomplishes the same through micro-batching, fast execution of batches on small sets of data.

8.2. Apache Flink

Apache Flink is an open-source stream processing framework that processes streams in real-time [15]. This is almost similar to Apex but not as widely used.

8.3. Apache Storm

This is a free and open source distributed real-time computation system [16]. This is fast but not stateful like Apache Apex.

8.4. Apache Samza

Apache Samza is a distributed stream processing framework [17].It was first developed by LinkedIn and later opensourced [18]. It is built on top of Apache Kafka [19], a distributed streaming platform. It provides stateful streaming capabilities [18]. It has great compatibility with Kafka [18]. Streaming applications can be built in such that Kafka consumes the data processed by Samza [18].

9. USERS

With its stream processing capabilities, Apache Apex facilitates building large scale real-time analytics applications. Enterprises like GE, PubMatic, SilverSpring Networks are using Apex based streaming solutions [9].

10. CONCLUSION

Apache Apex is an open source YARN(Hadoop 2.0)-native platform [6]. It unifies stream and batch processing. It can be used for processing both streams of data and static files making it more relevant in the context of present day internet and social media. It is aimed at leveraging the present Hadoop platform and reducing the learning curve for development of applications over it. It is aimed at It can used through a simple API. It enables reuse of code by not having to make drastic changes to the applications by providing interoperability with existing technology stack. It leverages the existing Hadoop platform investments.

ACKNOWLEDGEMENTS

This paper has been written as part of a class assignment for the course: I524: Big Data Software and projects, Spring 2017, School of Informatics and computing, Indiana University, Bloomington. Special thanks to Professor Gregor von Laszewski, Dimitar Nikolov and all associate instructors for guiding through the process of writing this paper.

REFERENCES

[1] Apache, "Apache software foundation blog apex introduction," Web Page, Apr. 2015, accessed: 2017-03-26. [Online]. Available: https://blogs.apache.org/foundation/entry/the_apache_software_foundation_announces90

[2] A. Kekre, "Apache apex blog incubator," Web Page, Sep. 2015, accessed: 2017-03-26. [Online]. Available: https://www.datatorrent.com/blog/apex-accepted-as-apache-incubator-project/

[3] ——, "Apache apex blog introduction," Web Page, Sep. 2015, accessed: 2017-03-26. [Online]. Available: https://www.datatorrent.com/blog/introducing-apache-apex-incubating/

[4] Apache, "Apache apex," Web Page, 2016, accessed: 2017-03-26. [Online]. Available: https://apex.apache.org/

[5] ——, "Apache apex documentation," Web Page, 2016, accessed: 2017-03-26. [Online]. Available: https://apex.apache.org/docs/apex/application_packages/#apache-apex-packages

[6] Wikipedia, "Apache apex wiki," Web Page, accessed: 2017-03-26. [Online]. Available: https://en.wikipedia.org/wiki/Apache_Apex

[7] Apache, "Apache apex application development documentation," Web Page, 2016, accessed: 2017-03-26. [Online]. Available: https://apex.apache.org/docs/apex/application_development/

[8] ——, "Apache apex application operator documentation," Web Page, 2016, accessed: 2017-03-26. [Online]. Available: https://apex.apache.org/docs/apex/operator_development/

[9] T. Weise, "Apache apex slideshare," Web Page, Jul. 2016, accessed: 2017-03-26. [Online]. Available: https://www.slideshare.net/ThomasWeise/apache-apex-stream-processing-architecture-and-applications

[10] Apache, "Apache apex application package documentation," Web Page, 2016, accessed: 2017-03-26. [Online]. Available: https://apex.apache.org/docs/apex/application_packages/#apache-apex-packages

[11] ——, "Apache apex introduction slideshare," Web Page, Jul. 2016, accessed: 2017-03-26. [Online]. Available: https://www.slideshare.net/ThomasWeise/apache-apex-stream-processing-architecture-and-applications

[12] ——, "Apache apex cli documentation," Web Page, 2016, accessed: 2017-03-26. [Online]. Available: https://apex.apache.org/docs/apex/operator_development/

[13] S. Hall, "The newstack article on apache apex competition," Web Page, May 2016, accessed: 2017-03-26. [Online]. Available: https://thenewstack.io/apache-gets-another-real-time-stream-processing-framework-apex/

[14] Apache, "Apache spark," Web Page, 2016, accessed: 2017-03-26. [Online]. Available: http://spark.apache.org/

[15] S. Hall, "The newstack article on apache flink," Web Page, Apr. 2016, accessed: 2017-03-26. [Online]. Available: https://thenewstack.io/apache-flink-addresses-continuous-stream-processing/

[16] Apache, "Apache storm," Web Page, 2016, accessed: 2017-03-26. [Online]. Available: http://storm.apache.org/

[17] ——, "Apache samza," Web Page, 2016, accessed: 2017-03-26. [Online]. Available: http://samza.apache.org/

[18] S. Hall, "The newstack article on apache streaming projects," Web Page, Jun. 2016, accessed: 2017-03-26. [Online]. Available: https://thenewstack.io/apache-gets-another-real-time-stream-processing-framework-apex/

[19] Apache, "Apache kafka," Web Page, 2016, accessed: 2017-04-06. [Online]. Available: https://kafka.apache.org/