Demo: Scalable Complex Event Processing on a Notebook

Miyuru Dayarathna, Minudika Malshan, Srinath Perera, Malith Jayasinghe
WSO2, Inc.
787, Castro Street
Mountain View, CA, USA 94041
{miyurud,minudika,srinath,malithj}@wso2.com

ABSTRACT

Recently data analytics notebooks are becoming attractive tool for data science experiments. While data analytics notebooks have been frequently used for batch analytics applications there are multiple unique problems which need to be addressed when they are used for online analytics scenarios. Issues such as mapping the event processing model into notebooks, summarizing data streams to enable visualizations, scalability of distributed event processing pipelines in notebook servers remain as some of the key issues to be solved. As a solution in this demonstration we present an implementation of event processing paradigm in a notebook environment. Specifically, we implement WSO2 Data Analytics Server (DAS)'s event processor in Apache Zeppelin notebook environment. We first demonstrate how an event processing network could be implemented in a stream processing notebook itself. Second, we demonstrate how such network could be extended for distributed stream processing scenario using WSO2 DAS and Apache Storm. Also we discuss about various improvements which need to be done at the user interface aspects to develop stream processing network in such notebook environment.

CCS CONCEPTS

• **Information systems** → **Data streams;** • **Human-centered computing** → **Visualization systems and tools;** • **Applied computing** → *Event-driven architectures;*

KEYWORDS

Data Analytics, Notebook, Stream Processing, Complex Event Processing

ACM Reference format:
Miyuru Dayarathna, Minudika Malshan, Srinath Perera, Malith Jayasinghe. 2017. Demo: Scalable Complex Event Processing on a Notebook. In *Proceedings of DEBS '17, Barcelona, Spain, June 19-23, 2017,* 4 pages.
DOI: 10.1145/3093742.3095093

DEBS '17, Barcelona, Spain
 978-1-4503-5065-5/17/06...$15.00
DOI: 10.1145/3093742.3095093

1 INTRODUCTION

Big data analytics is a complicated process which involves multiple steps such as data gathering, data storage, cleaning, model/algorithm building, visualizing, communication, etc. According to a recent report the accessibility for data and analytics within organizations are reducing dramatically because democratization of data analytics creates the potential for an explosion of creative thinking [6]. Data analytics pipelines play a pivotal role in this context. Data analytics notebooks are becoming popular among data analysts as a means of constructing, managing, and executing interactive, exploratory data science pipelines in an efficient manner. Notebooks extend the console-based approach for interactive computing by providing a Read-eval-print loop (REPL) based web application suitable for conducting the whole computation process: developing, documenting, executing code, and communicating the results [7]. Notebooks provide multiple advantages for data scientists over the traditional data analysis user interfaces (UIs) which makes them attractive tool for interactive data analysis. First, they provide the ability of revisiting a data analysis process multiple times by storing the intermediate results inside the notebook. They can be closed and reopened with the previous analysis results. Second, most of them provide the ability of interacting with multiple data analysis tools within one notebook constructing complex data analysis pipelines with several data analysis tools/frameworks. This provides the added benefit of leveraging the inherent characteristics associated with individual tool/framework to construct and experiment with complex data analysis scenarios in an interactive manner.

Traditional use cases for notebooks have been batch analytics. However, recent batch analytics data flows have been complimented by streaming (online/real-time) analytics as well as predictive (i.e., using Machine Learning) pipelines. Furthermore, some of the notable batch data processing systems have integrated stream processing capabilities within the system itself. For example, Apache Spark and Apache Flink frameworks have capabilities of specifying both batch and streaming analytics pipelines. Since data analytics notebooks have been originally developed targeting batch analytics scenarios, they remain completely alien for stream processing scenarios. Several problems exist in introducing stream processing model of execution to data analytics notebooks. First, the method and techniques to follow visualizing event streams and the operations carried out during the execution of an event processing network remains a significant challenge. While there have been previous attempts of visualizing event streams in tabular forms in spreadsheets, several challenges

exist when using the tabular form of stream representation in notebooks. Second, techniques need to be explored how a big data event processing pipeline could be orchestrated in a notebook environment. For example, suitable visualization techniques need to be followed which utilizes proper event summarization.

As a solution in this demonstration we present an event processing notebook system which explores how to effectively address the above-mentioned issues. Our notebook consists of three types of segments which could be used to construct an end to end scenario of an event processing network. These are called *Data Receiver*, *Query Processor (Siddhi)*, and *Data Publisher*. The Data Receiver is responsible for making data streams available for the streaming query networks developed in the notebook. Query Processor consists of stream processing queries (specified in Siddhi query language). The Data Publisher is the component which does publishing the event stream to external subscribers. We demonstrate how effectively a stream processing application can be developed in our notebook environment by implementing a financial trading application. Furthermore, we demonstrate how effectively we could interact with other data analysis systems/frameworks using WSO2 Data Avalytics Server (DAS) on our notebook environment.

The rest of the contents of this paper appear as follows. The related work are listed in Section 2. System design of event processing notebook is presented in Section 3. We describe the implementation details in Section 4. We provide an overview to the demonstration in Section 5 and conclude the paper in Section 6.

2 RELATED WORK

READ-EVAL-PRINT-LOOP (REPL) is a concept which evolves from LISP programming language which creates the basis for data analytics Notebooks [8]. The REPL architecture basically processes a sequence of commands and reports results immediately to the user. READ operation processes the syntax of a given expression and internalizes the expression as a semantic operation on the program state. EVAL runs the internalized expression and updates the top-level state accordingly. PRINT outputs the results of the evaluation. LOOP continues the READ,EVAL,PRINT infinitely until the user terminates the command interpreter. One drawback of conventional REPL is unless the user explicitly saves her script/state to a file the entire session is lost when the REPL environment shutdowns [1]. However, in notebooks model every scripting action is implicitly persisted as it happens.

Another recently proposed yet heavily influential theme for notebooks is called Live Programming [2]. This concept enables more fluid problem solving compared to "edit-compile-debug" style programming. One of the prominent examples for Live Programming is spreadsheets. Data and formulae can be edited and the effects of those edits can be observed immediately with spreadsheets [2].

There have been several attempts for constructing stream processing Graphical User Interfaces (GUIs) previously. In one such works Hirzel *et al.* discussed the usage of spreadsheet processors as tools for programming stream processing applications [3]. They mentioned that while the familiar interface and computation model makes it possible for domain experts to participate in the development of Complex Event Processing (CEP) applications natural operations found in streaming applications such as windowing and partitioning are not directly expressible in spreadsheets due to the spreadsheet's boundedness. However, this work had limitations of handling dynamic-sized windows and partitions. For example, with the 2D-spreadsheets model, in a scenario of processing a stream of stock quotes, one can partition the stock stream by symbol allocating a different sheet for each symbol. However, this becomes cumbersome when the number of stock symbols are large and if all the symbols are not known before. To overcome this issue Hirzel *et al.* introduced an improved stream processing spreadsheet model where an individual cell can represent a time-based window which is variable in size and specified using a duration [4]. In this approach user can specify the partitioning criterion while the spread sheet shows a debug view of a particular key while the server conducts the same computation separately for all the keys. This has enabled processing variable-sized and unbounded data sets which are common to time-based windows and partitions. Different from these works in this work our focus is on notebooks which are web applications with different type of data organization.

IPython Notebooks are one of the earliest examples for data analytics notebooks which originated in 2011 [7]. Currently they are the most widely used notebooks with more than fifty kernels and more than twenty well known organizations including Google, Microsoft, and IBM using them.

i2 Analyst's Notebook [5] developed by IBM is a unique notebook application which is commercially available in general and premium editions. A key feature of this application is its complete GUI based data analysis functionality [5].

3 SYSTEM DESIGN

We have designed stream processing notebook by generalizing the stream processing operations into three sub-modules as Receiver, Siddhi, and Publisher. Furthermore, the system has been designed for both non-distributed as well as distributed scenarios. System design of the stream processing notebook is shown in Figure 1.

3.1 Data Receiver

Data Receiver (i.e., Receiver) is the component designed to receive input streams. Receiver provides functionality to create input event streams and provide input event data which need to be processed. Receiver generates event data as text or from a Comma Separated Values (CSV) file. It also allows for creation of streams according to the nature of event data.

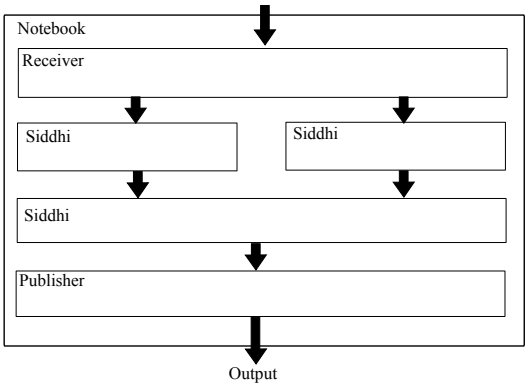

Figure 1: Stream Processing Notebook System Design.

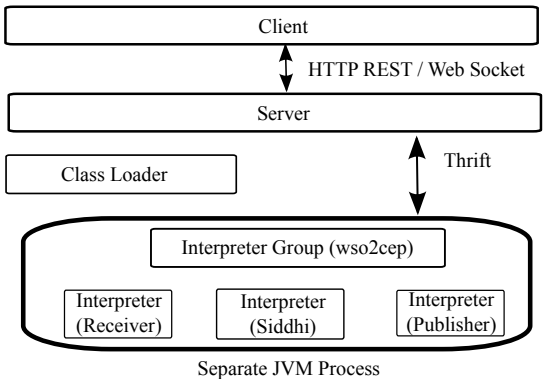

Figure 2: Stream Processing Notebook Implementation.

3.2 Query Processor

Query Processor(denoted as Siddhi) component conducts the processing of streaming data. We use the same term "Siddhi" to name this component although Siddhi in general refers to the CEP query language which runs on WSO2 DAS. This component takes care of running the execution plan with the input events and directing the output to an output stream which is defined within the execution plan. Multiple Siddhi components can be grouped together to create a complete execution plan as shown in Figure 1. Such organization allows us to use different types of execution backends with the stream processing notebook. While simple stream processing application prototypes can be developed with default Siddhi component, the same set of queries can be deployed in distributed cluster to run experiments with large amounts of streaming data.

3.3 Publisher

Publisher component provides facility to analyze output of the execution process by using Zeppelin's visualizing tools. In the current system design, the publisher accepts only single stream as its input and conducts visualizations of that stream. Hence, if multiple streams need to be visualized, they should be directed to multiple publishers or a Siddhi component should be used to merge streams from multiple publishers into single stream. In most of the time publisher acts as a visualization component. If the final output stream needs to be persisted it can be done with Siddhi component using the Siddhi's event tables.

4 IMPLEMENTATION

Our stream processing notebook has been implemented on top of Apache Zeppelin data analytics notebook. Each of the three sub-modules described in the above Section have been implemented as separate Zeppelin interpreters running inside the Zeppelin server. These interpreters have been grouped together as an interpreter group named *wso2cep* (See Figure 2).

The client is a web browser which communicates with the server via HTTP REST protocol.

5 DEMONSTRATION OVERVIEW

In our demonstration we allow the visitor to experience how an end to end stream processing application can be developed using our stream processing notebook. The demonstration includes the following components.

Receiver. First, we demonstrate the Receiver component which provides the functionality to create input event data streams. The Receiver allows for either specify a CSV file which contains the event data (See Figure 3) or to specify the events by directly typing their attributes (See Figure 4).

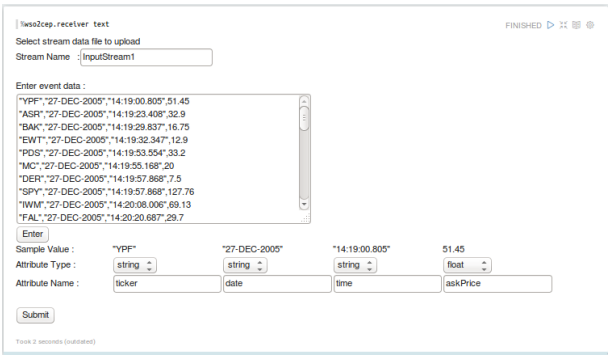

Figure 3: Creating a stream via uploading a CSV file.

Siddhi. Second, we demonstrate the Siddhi component which runs the stream processing query. Siddhi interpreter accepts the input stream created by the Receiver interpreter, executes the streaming operation (projection in Figure 5), and outputs the resulting event stream into another stream which is the output stream. We will also describe how a Siddhi query network can be deployed in Apache Storm via the stream processing notebook using Siddhi language's annotations.

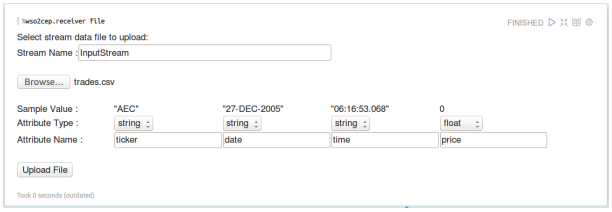

Figure 4: Creating a stream via directly typing the data.

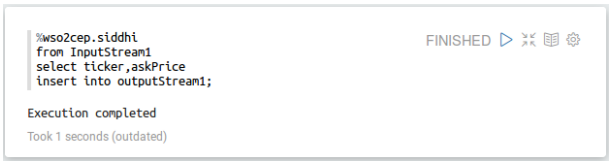

Figure 5: A projection query specified in Siddhi interpreter.

Publisher. Finally, we demonstrate the Publisher component which displays the output using Zeppelin's visualization tools. We will visualize the output sent from the Siddhi interpreter on the Publisher interpreter in different forms. This include different forms of visualization such as Tabular (Figure 6), bar chart (Figure 7), and pie chart (Figure 8).

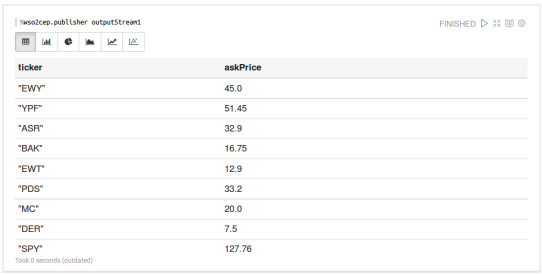

Figure 6: Visualization of output stream in tabular format.

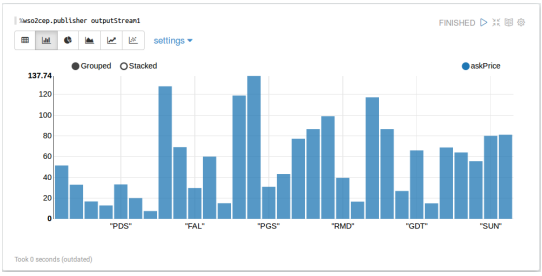

Figure 7: Visualization of output stream in bar chart format.

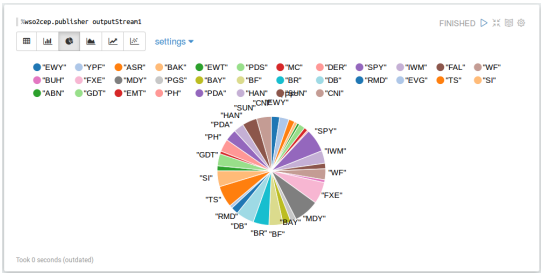

Figure 8: Visualization of output stream in pie chart format.

6 CONCLUSION

In this paper we demonstrate an implementation of a stream processing notebook on top of Apache Zeppelin. We designed and implemented the stream processing notebook as three interpreters which could be used to create multiple paragraphs linked together in such a way that it represents a stream processing application. We demonstrated the capabilities of the stream processing notebook using an interactive data stream processing application.

REFERENCES

[1] Mike Barnett, Badrish Chandramouli, Robert DeLine, Steven Drucker, Danyel Fisher, Jonathan Goldstein, Patrick Morrison, and John Platt. 2013. Stat!: An Interactive Analytics Environment for Big Data. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD '13)*. ACM, New York, NY, USA, 1013–1016. DOI: http://dx.doi.org/10.1145/2463676.2463683

[2] Sebastian Burckhardt, Manuel Fahndrich, Peli de Halleux, Sean McDirmid, Michal Moskal, Nikolai Tillmann, and Jun Kato. 2013. It's Alive! Continuous Feedback in UI Programming. In *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '13)*. ACM, New York, NY, USA, 95–104. DOI:http://dx.doi.org/10.1145/2491956.2462170

[3] Martin Hirzel, Rodric Rabbah, Philippe Suter, Olivier Tardieu, and Mandana Vaziri. 2015. Spreadsheets for Stream Partitions and Windows. In *Proceedings of the Second Workshop on Software Engineering Methods in Spreadsheets co-located with the 37th International Conference on Software Engineering (ICSE 2015)*. 39–40.

[4] Martin Hirzel, Rodric Rabbah, Philippe Suter, Olivier Tardieu, and Mandana Vaziri. 2016. Spreadsheets for Stream Processing with Unbounded Windows and Partitions. In *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems (DEBS '16)*. ACM, New York, NY, USA, 49–60. DOI:http://dx.doi.org/10.1145/2933267.2933607

[5] IBM Corporation. 2015. IBM i2 AnalystâĂŹs Notebook Premium. (2015).

[6] Incisive Media. 2015. Big Data Review 2015: A detailed investigation into the maturing of Big Data analytics. (2015).

[7] The IPython Development Team. 2015. The IPython Notebook. URL: https://ipython.org/ipython-doc/3/notebook/notebook.html. (2015).

[8] Makarius Wenzel. 2014. *Interactive Theorem Proving: 5th International Conference, ITP 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*. Springer International Publishing, Cham, Chapter Asynchronous User Interaction and Tool Integration in Isabelle/PIDE, 515–530. DOI:http://dx.doi.org/10.1007/978-3-319-08970-6_33