AutoStudio: A Generic Web Application for Transforming Dataflow Programs into Action

Nikhil-Kishor Rane, Omran Saleh

Technische Universität Ilmenau, Ilmenau, Germany
{first.last}@tu-ilmenau.de

Abstract:
In this paper, a user-friendly, interactive, and easy-to-use web-based application, is introduced. "AutoStudio" is a generic application where the users can generate dataflow programs for different dataflow languages, such as PipeFlow and Pig, using a drag and drop functionality. Using this application, the users can assemble the operators that comprise a particular program without being an expert in the target language and without his/her awareness of its operators and constructs. Therefore, the rapid development can be achieved. Our application also provides additional functionalities which makes it a "One-Stop-App" such as compiling, saving, executing, and checking the status of the generated dataflow programs.

1 Introduction

In today's world, an increasing amount of data is becoming obtainable over the internet. With the growth of systems focusing on the data processing, there is always a requirement of devising innovative ways of processing more data in less time, i.e., the big data challenge. Recently, several platforms have been developed to address this challenge. Typically, some of these platforms, e.g., PipeFlow [SBS15] and Pig[1], provide a declarative interface in the form of dataflow specifications where the processing steps are described by programs or scripts formulated as a sequence of statements. Each statement declares an operator which defines a specific task for manipulating the data, i.e., tuples. The flow of data is illustrated as lines or pipes between the operators. By using the output pipe of one operator as input pipe of another operator, a dataflow graph is formed.
However, the user currently has to manually perform all tasks of creating the dataflow script, editing, and compiling. Though, several text-based plug-in editors have been developed, e.g., pig-eclipse[2] and pig-mode[3] which are dedicated only for Pig[4] language, these editors focus mainly on syntax highlighting, code completion, and indentation. Namely, if the user needs to create and edit a particular dataflow script then he/she should be well versed with the syntax and programming constructs of the language. Therefore, no rapid development is possible as user needs to proceed by writing each programming statements

[1] http://pig.apache.org
[2] https://code.google.com/p/pig-eclipse/
[3] https://github.com/motus/pig-mode
[4] PipeFlow did not have any editor

correctly. Additionally, in these conventional approaches, concurrency is not supported and many users will not be able to simultaneously use the application. Moreover, the user also needs to manually check the status of programs in execution, which becomes cumbersome as the number of programs increases. This manual approach could lead to time and effort wastage if any errors or misconfigurations occur. These limitations motivated us to design a generic application to automate the whole process of generating and executing different dataflow languages; and as a result, **AutoStudio** was implemented.

2 AutoStudio System

The platform that we envision should be used by different people and organizations. It needs to be deployed only once and each user can access the application, create, and execute different dataflow scripts, such as PipeFlow and Pig, remotely from any location. With the advent of Internet, web application development has gained popularity. Web applications, presently, started substituting desktop applications for reasons of portability and usability. Thus, we have developed the **AutoStudio**. It is a very user friendly, interactive, and easy to use web application to run cross-platform using HTML5, Draw2D touch[5], and node.js[6]. It also relies heavily on JavaScript Object Notation (JSON) for data exchange and pre-compiled templates "hogan.js"[7]. AutoStudio's services and components are mentioned in Sect. 3 and Sect. 4, respectively. To perform any task, a user has to login to AutoStudio with a user ID and password. Once the user is successfully logged in, the user's home page is displayed with a list of **apps** and **flow-designs** sorted in the order they were last accessed as shown in Figure 1. This is the point where the **workflow** begins. Below is a brief description of the terms which make up AutoStudio.

Applications or "Apps": App is AutoStudio's terminology of referring to the design part of a workflow. An app defines a set of operators, connection types, properties, parameters, etc., which have pre-defined semantics. The semantics are later applied to a flow-design to transform it into a target language script. For instance, the apps which focus on PipeFlow and Pig scripts are PipeStudio and PigStudio, respectively.

Flow-design: The diagram a user creates in AutoStudio which is later transformed into a script is known as a flow-design. A flow-design corresponds to a specific app having pre-defined semantics. AutoStudio allows CRUD[8] operations on a flow-design and stores it primarily as JSON on the server. However, it can also be exported in JSON, SVG or PNG formats from AutoStudio at the click of a button.

Workflow: The point at which a user launches an app and starts creating a flow-design to the point when it actually executes on the server is called as workflow.

[5] `http://www.draw2d.org/draw2d/index.html`
[6] `http://www.nodejs.org`
[7] `http://twitter.github.io/hogan.js/`
[8] Create, read, update, and delete

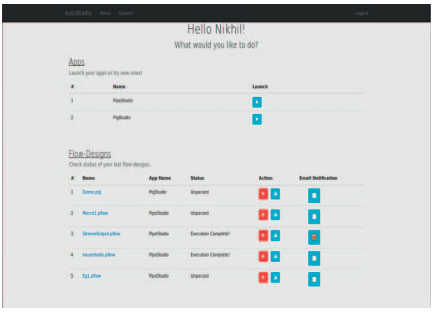

Figure 1: AutoStudio home page

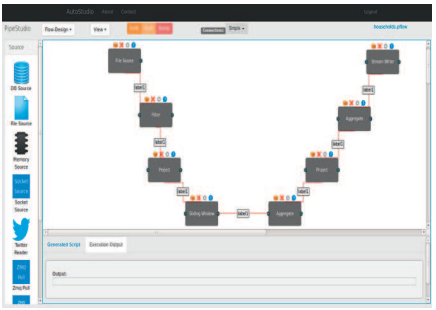

Figure 2: AutoStudio dashboard

3 AutoStudio Services

AutoStudio application provides several functionalities which makes it a "One-Stop-App" to handle the complete workflow. It enables users to leverage the emerging dataflow languages graphically via a collection of operators which could be "dragged and dropped" onto a drawing canvas (as in Figure 2). These operators are represented by icons. The user can assemble the operators in order to create a dataflow graph for a particular language in a logical way and visually show how they are related, and from this graph, equivalent script can be generated. This makes the user to be not aware of the language syntax and its constructs. Based upon the user's selection of apps from the home page, the right operators will be shown. By clicking on the operator icon, a pop-up window appears to let the user specify the parameters of operators, which are required. Moreover, the user can display the help contents for each operator. Besides these functionalities, the application also has a feature of script execution via calling the respective engine and displaying script execution result instantly and in real-time as well as emailing the user when the execution is complete. The application provides the options of saving the generated scripts or flow-designs for future reference, loading the saved script, and executing it whenever required.
Adding new operators: AutoStudio relies on JSON data for construction of its apps. Each app in our application is modeled in a single JSON file. Therefore, adding an operator, connection type or a container involves only adding the required {key:value} pairs in the app's JSON file. In our example of PipeStudio, this file is called *pipestudio.json*.
Adding new apps: Since our application is designed to be generic, a few components are needed in order to add a new app (currently, these components can be added manually, later, an interface can be used to configure the new app). These components are as follows:
1. *JSON file*: contains all configurations of operators, connection types, and containers.
2. *Resources*: all the required resources for the app like images, properties dialogs, etc.
3. *Database*: a database entry is required to make AutoStudio aware of this new app. The flow-designs created using AutoStudio are stored in separate database collections. Hence, each app has its own collection which provides flexibility to apps to handle their flow-designs as per requirement. 4. *Parser*: to parse Draw2D touch output into a target language. For instance, in PipeStudio, the parser parses the output into PipeFlow language.
5. *Executor*: a shell script or command to run the parsed dataflow script.

4 AutoStudio Components

AutoStudio prominently uses open source softwares and frameworks. It should work out of the box in any modern browser on any platform supporting HTML5. The components are divided into two parts. The first part is the client side which uses HTML5, JavaScript, Cascading Style Sheets (CSS), jQuery (and helping libraries), twitter bootstrap, and hogan.js. Mostly, they are used for building the graphical user interface, performing Ajax requests, file uploading and downloading, etc. AutoStudio extensively uses pre-compiled hogan templates where the data returned from the server is simply passed to these templates for quick rendering. In addition, Draw2D touch is used to enable creation of diagram applications in a browser by creating and manipulating operators and connections.
The second part is the server side which consists of the node.js web server and a database. The logic on the server side is completely implemented using node.js and its supporting modules. The event-driven, non-blocking I/O model of node.js makes it fast, light-weight, and efficient. It is greatly suitable for data-intensive real-time applications. Several node.js modules are used. *Nconf* is used as an object-store for easy storage and retrieval of configuration properties for AutoStudio, its apps, parsers, etc. *Nodemailer* is used to send emails from node.js. Our application uses this module to send notifications to registered email IDs when the execution of a flow-design is complete. *Socket.io* is a real-time framework server for node.js and used extensively to send real-time stats to the client when a flow-design is in execution. On the database side, the most popular No-SQL database *MongoDB* is used. It is an open source, JSON-style document oriented, and supports conventional as well as additional properties for data-intensive applications. It is used to store and retrieve the users and the saved flow-designs information.

5 Demonstration

During the demo session we will bring a computer running our application and demonstrate its capabilities, ease-of-use, generics, and supported-services as above-mentioned by employing real examples. The presenter will show how the users can navigate through different apps in AutoStudio. He will create different dataflow programs for different languages by dragging and dropping the operators. Furthermore, he will compile and execute these scripts by using our application. Thus, the audience can see real-time results and check the status of the running programs. Additional functionalities also will be shown such as saving and loading the scripts as well as emailing the user when the execution is complete. Moreover, the presenter will demonstrate how to extend any of the available apps, e.g., using PigStudio or PipeStudio as backend, and add new operators on-the-fly.

References

[SBS15] Omran Saleh, Heiko Betz, and Kai-Uwe Sattler. Partitioning for Scalable Complex Event Processing on Data Streams. In *New Trends in Database and Information Systems II*, pages 185–197. 2015.