Toward an API-Driven Infinite Cyber-Screen for Custom Real-Time Display of Big Data Streams

Mirco Soderi, Vignesh Kamath, John G. Breslin
Data Science Institute, National University of Ireland Galway, Ireland {firstname.lastname}@nuigalway.ie

2022 IEEE International Conference on Smart Computing (SMARTCOMP) | 978-1-6654-8152-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/SMARTCOMP55677.2022.00036

***Abstract*—Graphical User Interfaces (GUI) and real-time interactive Big Data charts play a key role in a wide variety of Big Data applications. The software libraries that are available at today are not suitable for displaying huge volumes of data in a single chart, because they require all the data to be collected at a single node. In this work, an innovative approach to the problem is presented, that consists in using a network of cyber-devices that is created and configured via API calls and that interfaces with a Scala Spark server application through a multiplicity of communication technologies, to produce and display a variety of time-space-infinite Big Data stream visualizations, including line plots, pie charts, histograms, that are updated at real-time as new data come, without ever collecting the data or the charts markup at a single node. The proposed approach characterizes for being (i) Web-based, (ii) API-based, (iii) Cloud-based, (iv) portable, (v) customizable/extendable, (vi) plug and play, and for relying on (i) Node-RED, (ii) MQTT, (iii) Scala, (iv) Akka HTTP, (v) Spark, (vi) Kafka, (vii) Docker. Remarkably, the same network used for Big Data visualization can be reconfigured in a matter of milliseconds and used for Big Data (streams) filtering, transformation, merge, analytics, and for training Machine Learning models, storing trained models on a Cloud storage, using stored models for one-shot or stream predictions, and much more. Although being at an advanced stage, we consider this research as a work in progress, since an extensive benchmarking and application to variegated real-world scenarios are still to be carried out.**

***Index Terms*—Web, Graphical User Interface, Dashboard, Plug and play, Big Data chart, Event stream, API, Micro-service, Portability, Docker, Scala, Spark, Kafka, MQTT, Node-RED**

I. Introduction

While opening to great opportunities, the increasing degree of automation in new generation industries, also said the Smart Factories [1], poses a number of compelling challenges. One of these is the need to rethink human-computer interfaces [2] to allow operators to timely and effectively supervise the production plant and take action in such a fast-changing environment. Graphical User Interfaces (GUI) and real-time interactive Big Data charts play a key role in this, but they need to be wisely integrated with the Cloud-based data storage and streaming technologies and tools [3] [4], for not to constitute a bottleneck.

In this work, a software architecture is presented that is aimed at displaying in a single chart, updated at real-time, arbitrary volumes of data flowing from event streaming servers. This is made possible by an innovative usage of Spark Structured Streaming [5] that allows to retain on the Cloud all the data that flow from the streaming server, and produce all the necessary HTML/SVG/JS markup still on the Cloud by means of tail recursive functions. The markup is then distributed to Web clients via socket. This way, charts are displayed without ever collecting or storing the dataset or the markup at any (high-resource) single node.

The proposed architecture characterizes for being (i) Web-based, that allows access from a variety of devices without any specific software; (ii) API-based, so that data sources, chart sizes, possible time windows, and much more, can be configured from remote in a matter of milliseconds by human operators and/or automated software; (iii) Cloud-based, that enables displaying potentially infinite volumes of data in every single data chart; (iv) portable, since all the components run in dedicated Docker containers; (v) customizable, since the Scala modules that produce the different types of chart are open-source, and potentially unlimited custom modules can be added to the Artificial Intelligence Server; (vi) plug and play, since it is not necessary to restart the Artificial Intelligence Server to load new modules or customize the existing ones.

An extensive review of the approaches and tools available at today for drawing Big Data charts has been conducted, and it was found that the (i) Swiftvis2 [6] and Vegas-viz libraries integrate with Spark through implicit conversions that encompass the collection of the whole dataset at the driver node, and (ii) Plotly, Breeze-viz, NSPL, scala-chart and WISP do not integrate with Spark, so data collection and transformation are once again needed. As a result, none of them is an option.

Remarkably, the architecture that is presented in this work can be today [9] expanded, reduced, reconfigured, in a matter of milliseconds, via API calls, to be converted to other usages.

The paper is structured as follows. The research question, state of the art, and newly proposed approach, are introduced in Section I. The architecture is described in Section II. The tasks that are currently implemented and ready to use for producing a variety of data charts are presented in Section III. A proof of concept is presented in Section IV. Conclusions are drawn in Section V.

II. Architecture

The complete software architecture is represented in Fig. 1. It includes both the core components that play a pivotal role in the chart production and distribution (represented on the right of the figure), and some auxiliary components (on the left) that

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant Number SFI/16/RC/3918 (Confirm).

2693-8340/22/$31.00 ©2022 IEEE
DOI 10.1109/SMARTCOMP55677.2022.00036

are very unlikely to be found in a production deployment, but that allow to develop, test, and showcase existing and new visualizations very conveniently.

The architecture essentially consists of a number of Service Nodes [8] represented as red boxes, and of an Artificial Intelligence Server (AIS) [9] represented as a cyan cloud.

A. Core components

In this paragraph, the architecture components are described with reference to the line plot. Anyway, the same apply to any stream visualization task, and most considerations even apply to any AIS stream task. Looking at the right side of Fig. 1, from top to bottom, we can see that API calls are made to the Service Node to configure, start and stop the chart production. The Service Node makes corresponding API calls to the AIS. Once started, the StreamVizLine task subscribes to the configured Kafka topic, and for each incoming message the updated markup is sent to the Service Node via Web socket, to be published to a configured MQTT broker and topic. The *html* Service Node listens to that topic and relies on node-red-contrib-dashboard for displaying the markup.

Designed to work in low-bandwidth situations, MQTT is now the leading open source protocol for connecting IoT devices. Kafka is the most popular and the only event streaming server supported by Spark.

B. Auxiliary components

Looking at the left side of Fig. 1, from bottom to top, the inject Service Node exposes a POST API and publishes the received bodies to a configured MQTT broker and topic. Through the broker, the data flow to a http Service Node, which makes an authenticated API call to write them to the file system of the AIS, in the portion reserved to the Service Node Θ.

Now, from top to bottom, API calls are made to configure and control the Service Node Θ. In particular, the configured task is "FileStream". This task monitors the file system and publishes the changes to a configured Kafka topic, that is the same from where the StreamVizLine task is configured to listens.

III. Artificial Intelligence Server Tasks

The task that is currently available and ready-to-use in the AIS for producing the markup for a line plot that displays a Big Data stream coming from a Kafka topic is named *StreamVizLine*. It generates a plot where the time is on the X-axis, the values are on the Y-axis, and a line is drawn for each key seen in the input stream. The example plot produced in the proof of concept described in Section IV is represented on the bottom right corner of Fig. 1.

A Spark Structured Streaming query with grouping, and a set of tail recursive functions, stand at the basis of the implementation, and allow to store all the data on the Cloud, and produce the markup for the chart from the raw data still on the Cloud. This is key to produce charts where data sets of arbitrary size are represented without relying on any high resource device. This same approach is adopted also for pie charts, and histograms.

When the StreamVizPie AIS task is used, a separated pie chart is displayed for each different key that is seen in the messages that come from the event streaming server, and a slice is depicted for each seen value. The size of the slice depends of how many times the value has been seen, with respect to the others bearing the same key. The pie chart produced in the proof of concept described in Section IV is represented in Fig. 2.

If the StreamVizHist AIS Task is used, a separated histogram is displayed for each different key that is seen in the messages that come from the event streaming servers. The number of the bins (columns) to be drawn in the histogram is accepted as a configuration parameter. The height of each column depends on how many values are seen that fall in the bin/column boundaries. The histogram produced in the proof of concept described in Section IV is represented in Fig. 3.

IV. Proof of concept

For setting up a proof of concept, refer to the GitHub repository[1] associated to this paper. Create a new user-defined Docker bridge/network, and name it ExampleNetwork. Then, create the following Service Nodes mapped to ports from 2130 onward: VizHtml, VizAI, VizStreamWriter, VizStreamInput, VizAIInput, VizAI2, VizAI2Input. Then, use the artifacts available in the GitHub repository for creating and running (i) a mozilla/sbt container named AISv2; (ii) Node-RED containers named BrokerACL, ServiceNodeACL, and TransformationLibrary, mapped to ports from 1990 to 1992; (iii) an emqx 4.3.2 container named ExampleBroker. Then, use the docker-compose.yml file to have a single node Kafka server up and running in your local Docker engine for demo and development purposes. All containers must be connected to the same Docker network. Finally, go through the Postman collection[2]. Connect to https://localhost:2130/ui to display the generated visualizations.

V. Conclusions

In this work, a software architecture for Big Data stream visualization as a Service has been presented. Motivations, involved technologies and tools have been presented in Section I. The architecture has been described in Section II. A variety of ready-to-use modules for composing a range of different data charts has been presented in Section III. A proof of concept has been presented in Section IV, including directions for creating the necessary Docker containers, configuring them, and making appropriate API calls to see the architecture in operation. In summary, it has been outlined a way to display Big Data in customizable charts refreshed at real-time and managed via API (so suitable for reconfigurable manufacturing [10]), without relying on high-resource nodes.

[1] https://github.com/mircosoderi/Toward-an-API-driven-infinite-cyber-screen-for-custom-real-time-visualizations-of-Big-Data-streams

[2] https://documenter.getpostman.com/view/16531967/UVeCQTfA

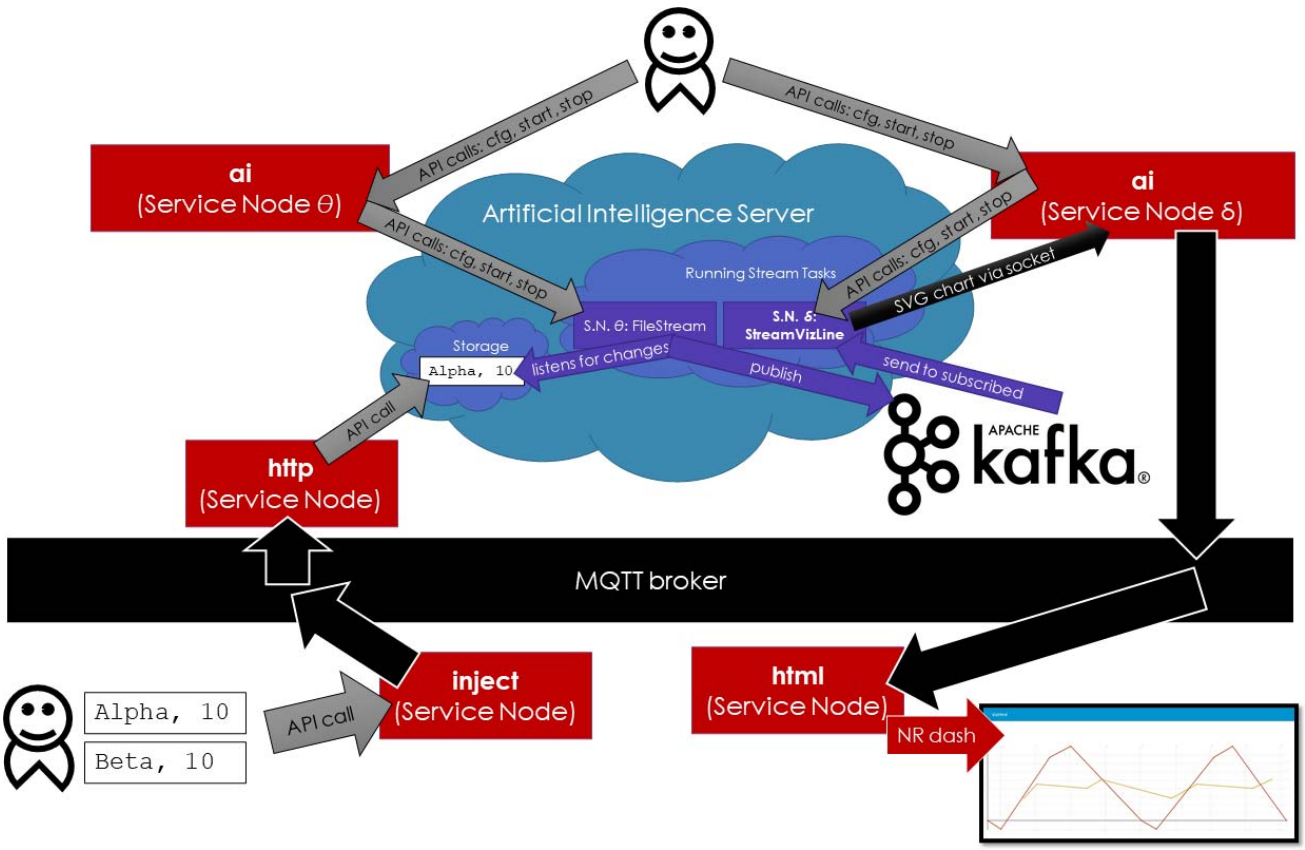

Fig. 1. The complete software architecture

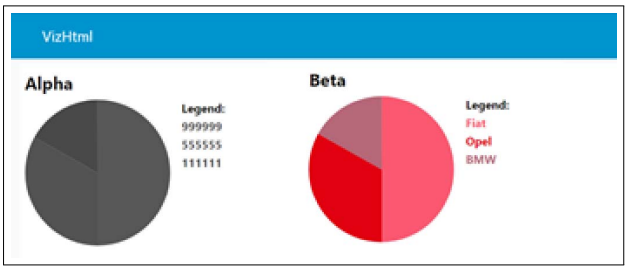

Fig. 2. Pie charts (proof of concept)

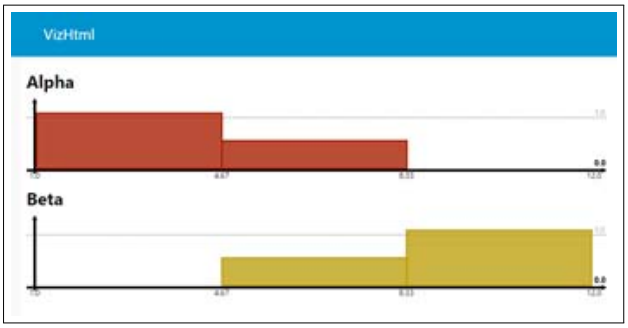

Fig. 3. Histograms (proof of concept)

REFERENCES

[1] Hozdić, Elvis. "Smart factory for industry 4.0: A review." International Journal of Modern Manufacturing Technologies 7.1 (2015): 28-35.

[2] Jakobs, Eva-Maria, et al. "Not ready for industry 4.0: Usability of CAx systems." international conference on applied human factors and ergonomics. Springer, Cham, 2017.

[3] Kanagachidambaresan, G. R. "Node-Red Programming and Page GUI Builder for Industry 4.0 Dashboard Design." Role of Single Board Computers (SBCs) in rapid IoT Prototyping. Springer, Cham, 2021. 121-140.

[4] Jwo, Jung-Sing, Ching-Sheng Lin, and Cheng-Hsiung Lee. "An Interactive Dashboard Using a Virtual Assistant for Visualizing Smart Manufacturing." Mobile Information Systems 2021 (2021).

[5] Armbrust, Michael, et al. "Structured streaming: A declarative api for real-time applications in apache spark." Proceedings of the 2018 International Conference on Management of Data. 2018.

[6] Lewis, Mark C., and Lisa L. Lacher. "Swiftvis2: Plotting with spark using scala." International Conference on Data Science (ICDATA'18). Vol. 1. No. 1. 2018.

[7] Leitner, Stefan-Helmut, and Wolfgang Mahnke. "OPC UA–service-oriented architecture for industrial applications." ABB Corporate Research Center 48.61-66 (2006): 22.

[8] Soderi, Mirco, et al. "Ubiquitous System Integration as a Service in Smart Factories." 2021 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS). IEEE, 2021.

[9] Soderi, Mirco, et al. "Advanced Analytics as a Service in Smart Factories" IEEE 20th International Symposium on Applied Machine Intelligence and Informatics (SAMI 2022). IEEE, 2022. (forthcoming)

[10] Bi, Zhuming M., et al. "Reconfigurable manufacturing systems: the state of the art." International journal of production research 46.4 (2008): 967-992.