(12)Indian Patent Application

(21) Application Number: 202141041151

(22) Filing Date: 14/09/2021 (43) Publication Date: 17/03/2023

(71) Applicant(s): L&T TECHNOLOGY SERVICES LIMITED

(72) Inventor(s): Kumbhalkar, Kaustubh
Dhangot, Shabbir Ismailbhai

(51) International Classifications: H04L 29/08 G06F 9/54 G06F 3/06 G06F 12/0815 G06F 16/84

(54) Title: METHOD OF DATA PROCESSING IN A DATA PIPELINE

(57) Abstract: A method of data processing is disclosed. The method may include data pipeline configuration. The method may include creating a job folder for a data pipeline. Setting the properties on the components dialogue from the frontend may also be part of the method. The method may also include transforming current data and delivering modified payload via the transformation component. Furthermore, the method may include inspecting all properties, files, and the data pipeline's validity. The method may also include initializing MQTT, storing its state, and passing it into other commands. The method may also include generating a Kafka topic and passing the Kafka topic to the application. properties, and then executing the application properties utilizing spark-submit command and storing the spark-submit process ID. The method may also include storing logic files in the backend files.

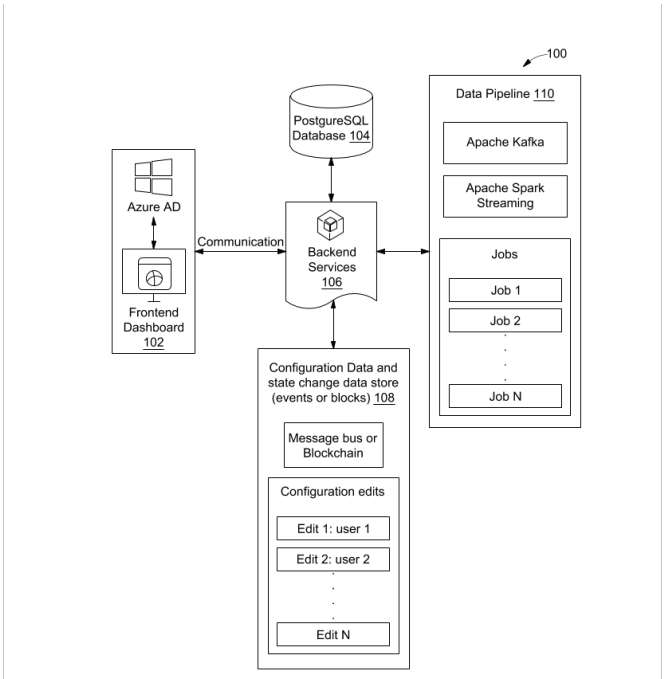

FIG. 1

-- Digitally Signed--

FORM 2

THE PATENTS ACT 1970
(39 OF 1970)
&
The Patent Rules, 2003

Complete Specification

(See Section 10 and Rule 13)

1. TITLE OF THE INVENTION

METHOD OF DATA PROCESSING IN A DATA PIPELINE

2. APPLICANT(S)

(a) NAME : **L&T TECHNOLOGY SERVICES LIMITED**

(b) NATIONALITY : **INDIAN**

(c) ADDRESS : **DLF IT SEZ Park, 2nd Floor – Block 3**

1/124, Mount Poonamallee Road,

Ramapuram, Chennai – 600 089,

INDIA.

3. PREAMBLE TO THE DESCRIPTION

COMPLETE

The following specification describes the invention and the manner in which it is to be performed

DESCRIPTION

Technical Field

[001] This disclosure relates generally to data pipeline, and a method of data processing in a data pipeline for storing a data pipeline change event in real-time.

BACKGROUND OF THE INVENTION

[002] A data pipeline is series of steps that ingest raw data from various sources and transport it to a storage and analysis location. In other words, a data pipeline is a set of data processing elements connected in series, such that an output of one element is the input of the next one. The elements of the data pipeline may be executed in parallel or in time-sliced fashion. The data pipeline is useful in industries and enterprise-level where there is data streaming from multiple channels and capturing this data is essential. However, there are many obstacles to clean data flow, for example, data corruption or multiple data sources producing conflicting or redundant information. The data pipelines may perform all the manual steps needed to solve a problem and turn the process into a smooth and automated workflow.

[003] Data streaming may be performed in a JSON (JavaScript Object Notation) formats which is a text-based standard for representing structured data that is based on JavaScript object syntax. However, storing of such data require huge space. Further, there are challenges in direct updating of the data, as database requires row data.

SUMMARY OF THE INVENTION

[004] In an embodiment, a method of processing data in a data pipeline is disclosed. The method may include data pipeline configuration. The method may include creating a job folder for a data pipeline. The method may also include setting the properties on the components dialog from frontend. The method may further include transforming current data and providing modified payload using transformation component. Furthermore, the method may include checking all properties, files and checking validity of the data pipeline. The method may also include initializing MQTT, storing its state and passing into another commands. The method may further includes generating Kafka topic, passing the Kafka topic to the application. properties, executing the application.properties using spark-submit command, and storing the spark-submit process ID. The method may further include storing logic files in the backend files.

BRIEF DESCRIPTION OF THE DRAWINGS

[005] The accompanying drawings, which are incorporated in and constitute a part of this disclosure, illustrate exemplary embodiments and, together with the description, serve to explain the disclosed principles.

[006] **FIG. 1** is a block diagram illustrating a data pipeline, in accordance with an embodiment of the present disclosure.

[007] **FIG. 2** is a process diagram showing activation of a job, in accordance with an embodiment of the present disclosure.

[008] **FIG. 3** illustrates a flowchart of a flow of data pipeline configuration, in accordance with an embodiment of the present disclosure.

[009] **FIG. 4** is another process diagram illustrating a flow of a data pipeline, in accordance with an embodiment of the present disclosure.

DETAILED DESCRIPTION OF THE DRAWINGS

[010] Exemplary embodiments are described with reference to the accompanying drawings. Wherever convenient, the same reference numbers are used throughout the drawings to refer to the same or like parts. While examples and features of disclosed principles are described herein, modifications, adaptations, and other implementations are possible without departing from the spirit and scope of the disclosed embodiments. It is intended that the following detailed description be considered as exemplary only, with the true scope and spirit being indicated by the following claims. Additional illustrative embodiments are listed below.

[011] Referring to **FIG. 1**, a block diagram of a system architecture 100 of a data pipeline application is illustrated, in accordance with an embodiment of the present disclosure. As shown in the FIG. 1, the system architecture 100 may include a frontend dashboard 102, a postgre SQL database 104, a backend service 106, and a configuration data or state change data store 108. Furthermore, a data pipeline 110 may be associated with the backend services.

[012] The frontend dashboard 102 may be coupled with azure active directory (AD). As will be appreciated, **Azure Active Directory** is a multi-tenant, cloud-based **directory** and an

identity management service of Microsoft™. The frontend may be is communicating with backend services 106 via “http” requests. For example, the frontend may send entered data to the backend. The backend might then again validate that data and finally store it in some database. The backend programming is done for creating a job folder for each data pipeline using node.js. A job folder may be created based on properties passed from the frontend. The properties are set on the components dialog from frontend. The components may include MQTT, Kafka, RDBMS. The frontend framework may be angular with basic functionality like drag-drop, grids, side menu for components that makes the process lightweight to load.

[013] The streaming data is ingested using framework like Azure Kafka and Azure Spark streaming. As will be further appreciated, Azure Kafka is a distributed, replicated messaging service platform that serves as a highly scalable, reliable, and fast data ingestion and streaming tool. It is high-throughput distributed messaging system in which multiple producers send data to Kafka cluster and which in turn serves them to consumers. It is a distributed, partitioned, replicated commit log service.

[014] The data in Kafka may be organized into *topics* that are split into *partitions* for parallelism. Each partition is an ordered, immutable sequence of *records*, and can be thought of as a structured commit log. Producers append records to the tail of these logs and *consumers* read the logs at their own pace. Multiple consumers can *subscribe* to a topic and receive incoming records as they arrive. As new records arrive to a partition in a Kafka topic, they are assigned a sequential identity (ID) number called an *offset*. A Kafka cluster retains all published records - whether they have been consumed - for a configurable retention period, after which they are marked for deletion.

[015] A large amount of data generated is processed by class of frameworks called Stream processing frameworks for example, Apache™ Storm, Apache™ Samza, and Apache™ Spark streaming. The Spark streaming is an extension of Spark Core which provides capabilities of fault tolerant processing of live stream data. The Spark streaming divides the incoming stream into micro batches of specified intervals and returns Dstream. The Dstream represents continuous stream of data ingested from sources like Kafka. The Dstreams are processed and pushed out to filesystems, databases, and live dashboards. A series of operations may be performed on the input DStream to obtain messages.

[016] When the workflow is executed, it is run on Apache™ Spark streaming and the results of execution of each of the jobs are streamed back to the user's web browser and displayed in JSON format. The workflow job is submitted to Apache™ Spark streaming. The driver collects the results of the distributed execution, converts it to JavaScript Object Notation (JSON) supported by PostgreSQL 104 and returns them back to the web server. The PostgreSQL 104 is an advanced, enterprise-class, and open-source relational database system, and supports both SQL (relational) and JSON (non-relational) querying. The web server in turn streams it back to the user's web browser where it is displayed in rich format using Java Script. A typical pipeline applies subsequent transforms to each new output until processing is complete. The transformation component may be used to transform current data and provide modified payload. The transformation may be supported by Java or Python script. A user interface allows the user to create and edit the workflows. It also allows the user to execute the workflows. The state of change and change in the configuration data are stored and can be recovered at any instant of time.

[017] Referring to **FIG. 2**, a process diagram 200 showing activation of a job is illustrated, in accordance with an embodiment of the present disclosure. Upon activation of the job, all the properties and files are checked. Further, validity of the data pipeline is examined. After examining, MQTT 202 may be initialized. The MQTT 202 may act as a server for receiving all messages from the clients and then routing the messages to the appropriate destination clients and store it state to pass into another commands. A state management may be performed at code level. The code will subscribe for MQTT event to pass it to another state.

[018] An IP may be connected to a MQTT host by using a host port. Each message to be published using MQTT may include a topic containing routing information for the host. A bridge may be customized between MQTT topic and Kafka topic which is unique and generated using pipeline ID that is used in MQTT properties 202. The Kafka topic may be passed to application.properties 204. The application.properties 204 may include details of a Relational Database Management System (RDBMS) 210 for publishing data. Further the application.properties 204 are executed using spark-submit 208 command which may further utilizes ubiqdatapipeline.jar file 206. After execution of the spark-submit process ID the spark.pid file and logs are stored in spark.logs. After saving in the logs the files are saved in the backend files for further investigation.

[019] Referring to **FIG. 3**, a flowchart 300 of a flow of data pipeline configuration is illustrated, in accordance with an embodiment of the present disclosure. At step 302, a data pipeline may be activated. For activation of the data pipeline, multiple commands are required to be executed. Backend services may validate all the information (process ids) and may run various commands to kill it at step 304. At step 306, process data pipeline may stop reading any more data from the streaming data. At step 308, files may be modified and job folder may be created for the data pipeline. The data pipeline may be created based on properties received from the frontend. The job may include mqtt.properties, application.properties, and connect-standalone.properties. The initializing of MQTT storing its state, passing into another commands and generating a Kafka topic using pipeline ID used in mqtt.properties may be carried out at step 310. At step 312, logs may be stored. The Kafka may pass topic to the application.properties, at step 314. The application.properties may be executed using spark-submit command, at step 316. The logs may be stored in the backend files for further investigation, at step 318.

[020] Referring to **FIG. 4**, a process diagram 400 illustrating a flow of a data pipeline is illustrated, in accordance with an embodiment of the present disclosure. When multiple changes are made to the data pipeline, it is critical to maintain and share a common state machine and information, which includes, in addition to the state, information about who and when the configuration change was made. This can be accomplished in a variety of ways, such as considering state machine as a design suitable for a lightweight user interface and application logic. When the user interface gets updated, the data pipeline may also get updated by the backend service. The backend service then updates the configuration data state change data store, and each configuration made by a user via the UI is updated in the database.

[021] At step 402, JSON configuration document for UI widgets for read-only and read-write access flag and widget identification may be updated, at step 402. At step 404, UI widget ID and JSON configuration pipeline element may be created. The common state as JSON document may be based on response from data pipeline configuration results. At step 406, editing of the data pipeline properties may take place, and all state changes of properties or commands of data pipeline may be saved at step 408. A database may store information for the tool's end user. When a user makes a change, it is broadcast over an event mechanism, and all user interfaces are updated with the current pipeline status. If there are changes, data pipeline gets initiated, at step 412. If no such edit is made by the user, then the data pipeline is stopped,

at step 410. Accordingly, at step 414, data pipeline running state may be checked. For example, MQTT and Apache Kafka are two event mechanisms that can be used for running state of data pipeline.

[022] The above disclosure provides a scalable technique for storing in real-time data pipeline configuration changes. Further, the techniques provide for a low-cost data pipeline tool with configuration change event and UI widget configuration change datastore. With improved performance, multiple data pipeline configurations can be stored and activated at the same time. A backend code logic saves configurations with logs for each pipeline, making it simple to debug. In the event of an error, rapid reversal and identification of changes can be made to bring into a stable state.

[023] It is intended that the disclosure and examples be considered as exemplary only, with a true scope and spirit of disclosed embodiments being indicated by the following claims.

WE CLAIM:

1. A method comprising:

creating a job folder for a data pipeline when the data pipeline is created based on properties received from a frontend, wherein the job comprises mqtt.properties, application.properties, and connect-standalone.properties;

setting the properties on the components dialog from frontend, wherein the components comprise MQTT, Kafka, RDBMS, wherein MQTT properties comprise host, port, protocol, topic, and wherein RDBMS properties comprise URL, db, username, password, type, table name;

transforming current data and providing modified payload using transformation component, wherein the transformation components comprise jar and python script;

upon activation of a job by a user, checking all properties and files and checking validity of the data pipeline;

initializing MQTT and storing its state and passing into another commands;

generating a Kafka topic using pipeline ID used in mqtt.properties, wherein the Kafka topic is unique;

passing the Kafka topic to the application.properties, wherein the application.properties comprises detail of RDBMS for publishing data;

executing the application.properties using spark-submit command, wherein the spark-submit uses ubiqdatapipeline.jar file;

storing spark-submit process ID spark.pid file and storing logs in sparks.logs; and

storing logic files in the backend file for any investigation to be done.

Dated this 08th Day of September 2022

-- Digitally Signed--
Bhanu Prasad
(INPA No: **3253**)
Manager, IPR Dept.,
L&T Technology Services Limited,
DLF 3rd Block, 2nd Floor,
Manapakkam, Chennai - 600089.

ABSTRACT

METHOD OF DATA PROCESSING IN A DATA PIPELINE

A method of data processing is disclosed. The method may include data pipeline configuration. The method may include creating a job folder for a data pipeline. Setting the properties on the components dialogue from the frontend may also be part of the method. The method may also include transforming current data and delivering modified payload via the transformation component. Furthermore, the method may include inspecting all properties, files, and the data pipeline's validity. The method may also include initializing MQTT, storing its state, and passing it into other commands. The method may also include generating a Kafka topic and passing the Kafka topic to the application. properties, and then executing the application properties utilizing spark-submit command and storing the spark-submit process ID. The method may also include storing logic files in the backend files.

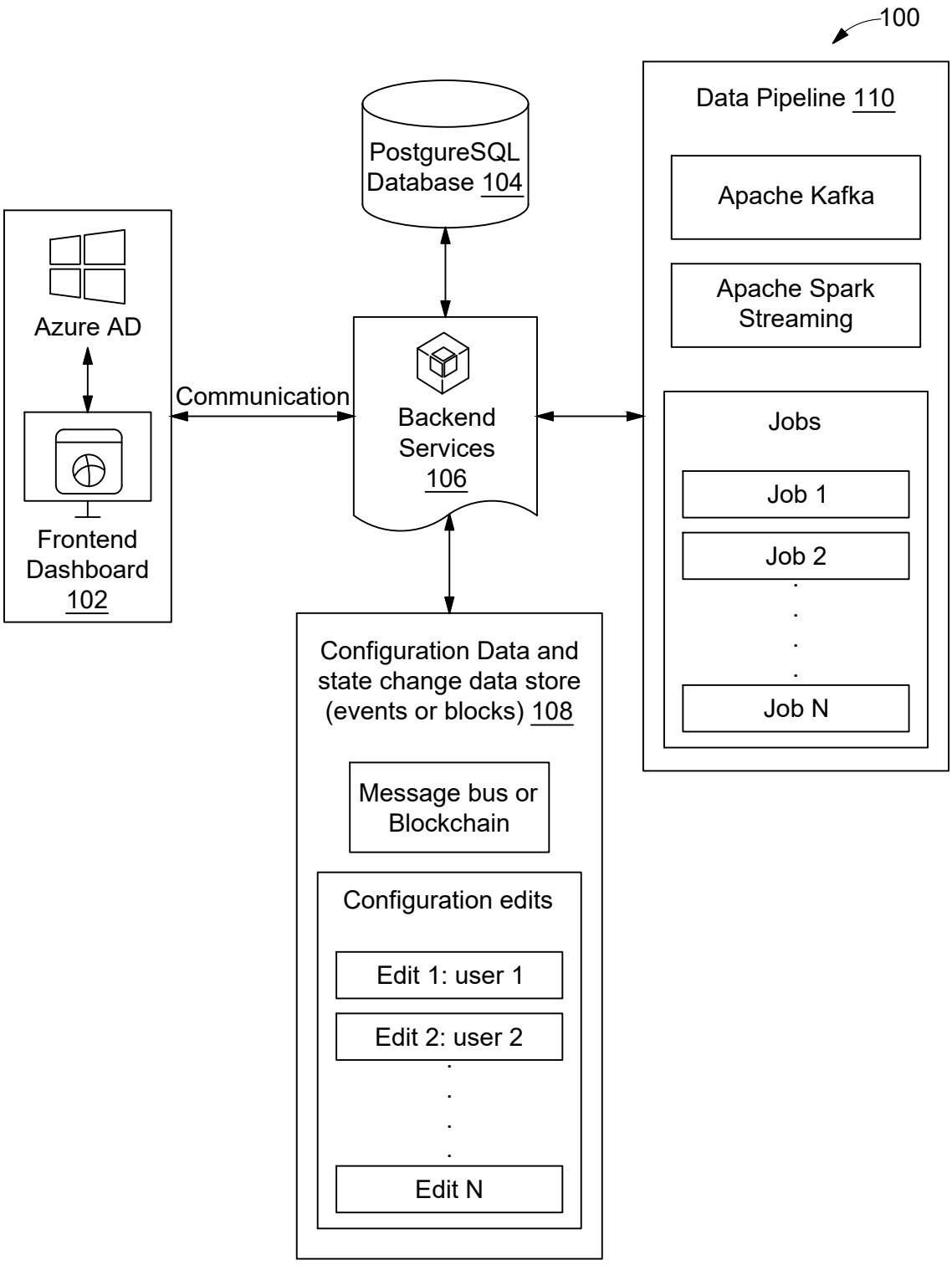

FIG. 1

-- Digitally Signed--
Bhanu Prasad
(INPA No: 3253)
Manager, IPR Dept.,
L&T Technology Services Limited,
DLF 3rd Block, 2nd Floor,
Manapakkam, Chennai - 600089.

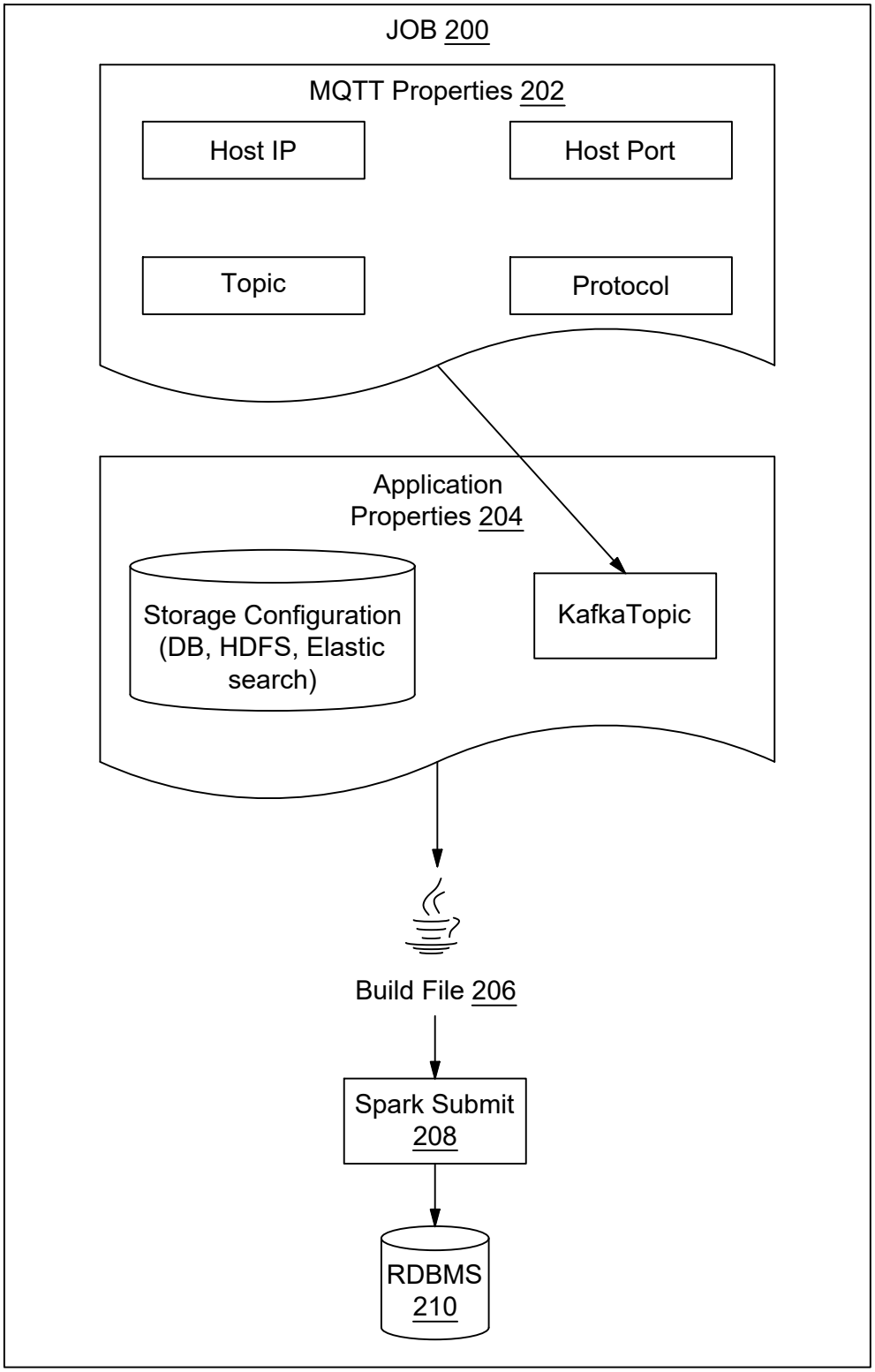

FIG. 2

-- Digitally Signed--
Bhanu Prasad
(INPA No: 3253)
Manager, IPR Dept.,
L&T Technology Services Limited,
DLF 3rd Block, 2nd Floor,
Manapakkam, Chennai - 600089.

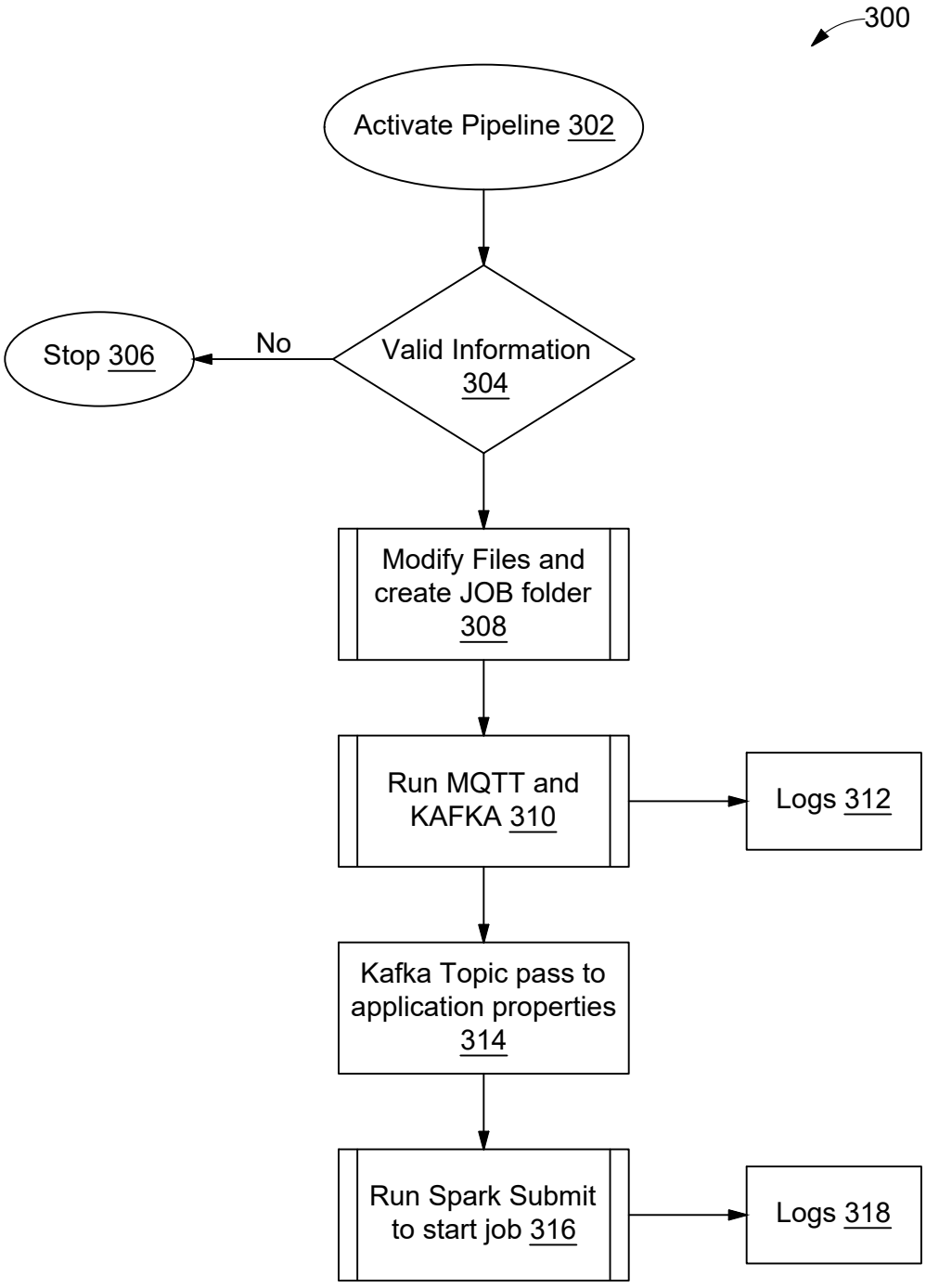

FIG. 3

-- Digitally Signed--
Bhanu Prasad
(INPA No: 3253)
Manager, IPR Dept.,
L&T Technology Services Limited,
DLF 3rd Block, 2nd Floor,
Manapakkam, Chennai - 600089.

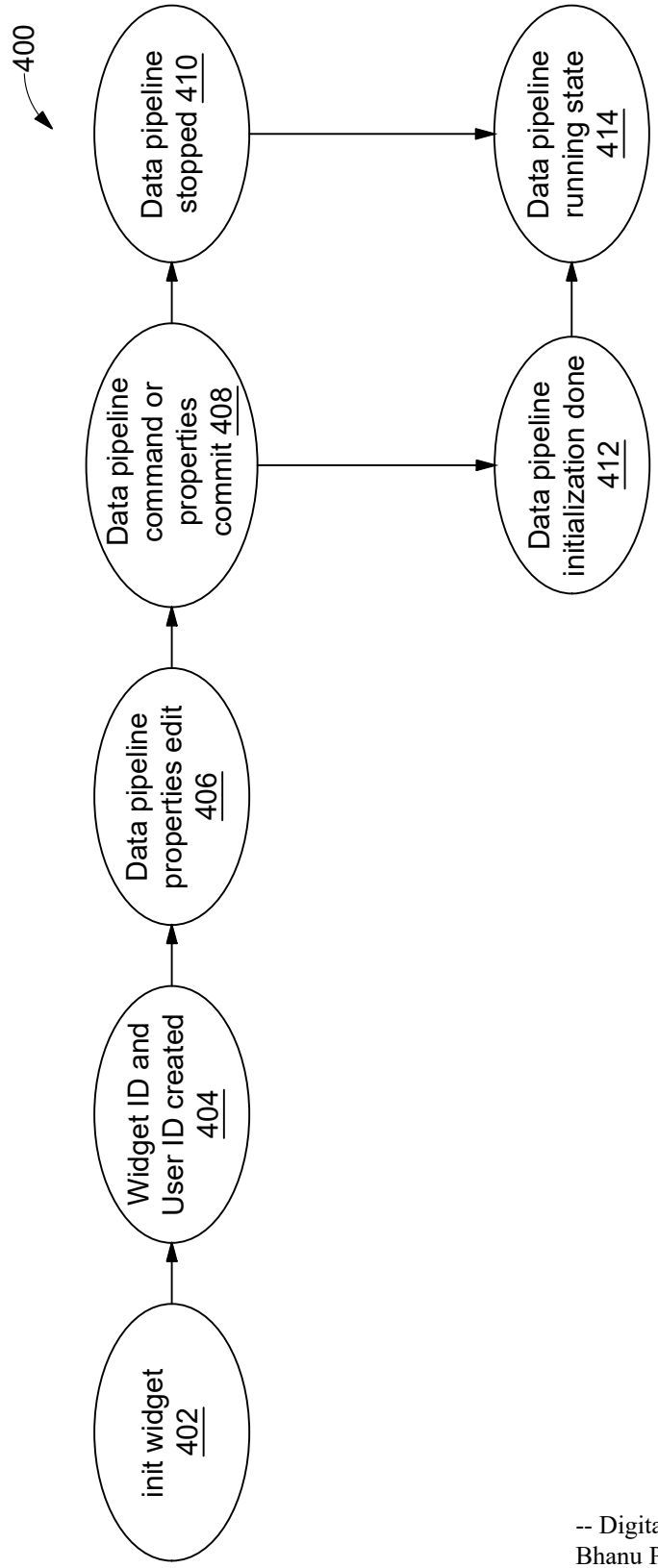

FIG. 4