US 20220100771A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2022/0100771 A1**

Pang et al. (43) **Pub. Date: Mar. 31, 2022**

(54) **AUTOMATIC TRANSFORMATION OF TIME SERIES DATA AT INGESTION**

(71) Applicant: **VMware, Inc.**, Palo Alto, CA (US)

(72) Inventors: **Clement Ho Yan Pang**, Sunnyvale, CA (US); **Lakshmi Ganesh N.R. Kapatralla**, Sunnyvale, CA (US); **Jason Hsi-Chieh Bau**, Palo Alto, CA (US)

(73) Assignee: **VMware, Inc.**, Palo Alto, CA (US)

(21) Appl. No.: **17/038,691**

(22) Filed: **Sep. 30, 2020**

Publication Classification

(51) **Int. Cl.**

G06F 16/25 (2006.01)

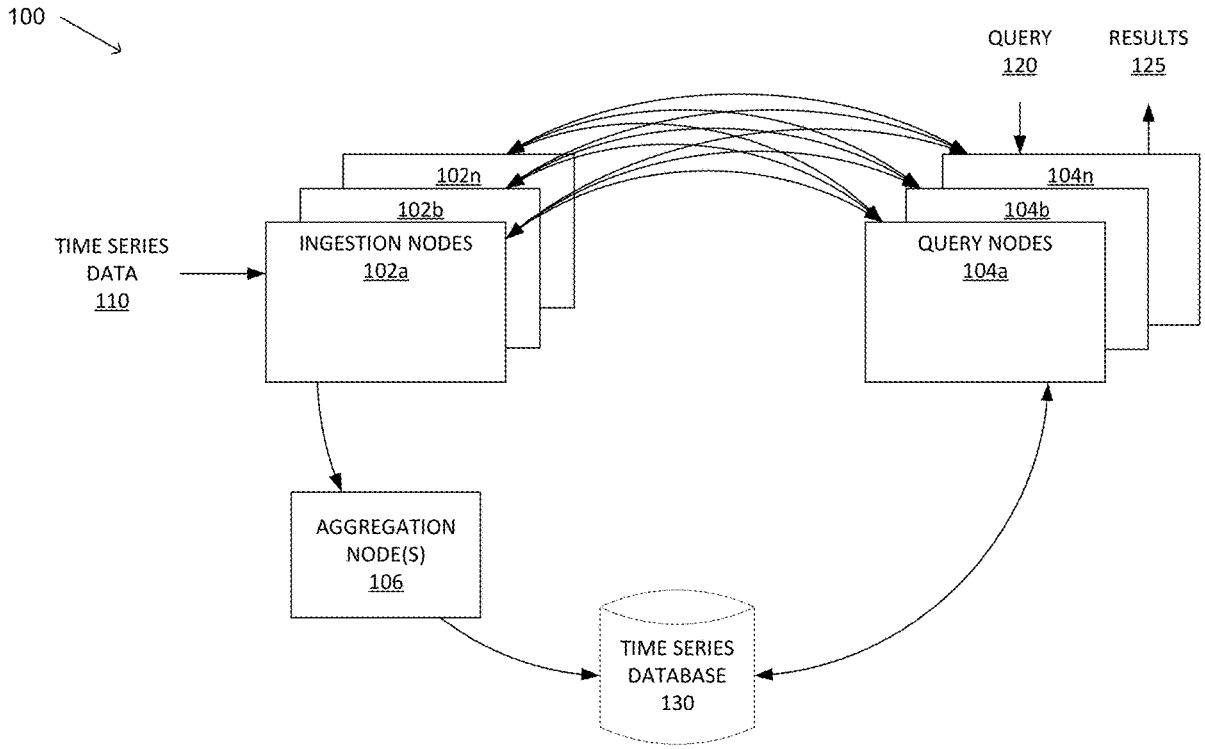
G06F 16/22 (2006.01)

(52) **U.S. Cl.**

CPC ***G06F 16/254*** (2019.01); ***G06F 16/2291*** (2019.01)

(57) **ABSTRACT**

In a computer-implemented method for automatic transformation of time series data at ingestion, time series data comprising data points is received at at least one ingestion node of a time series data monitoring system, wherein the data points have an input observability format. At the at least one ingestion node, the data points the data points are transformed from the input observability format to an output observability format according to configuration rules of the time series data monitoring system. The data points having the output observability format are forwarded from the at least one ingestion node to a persistent storage device.



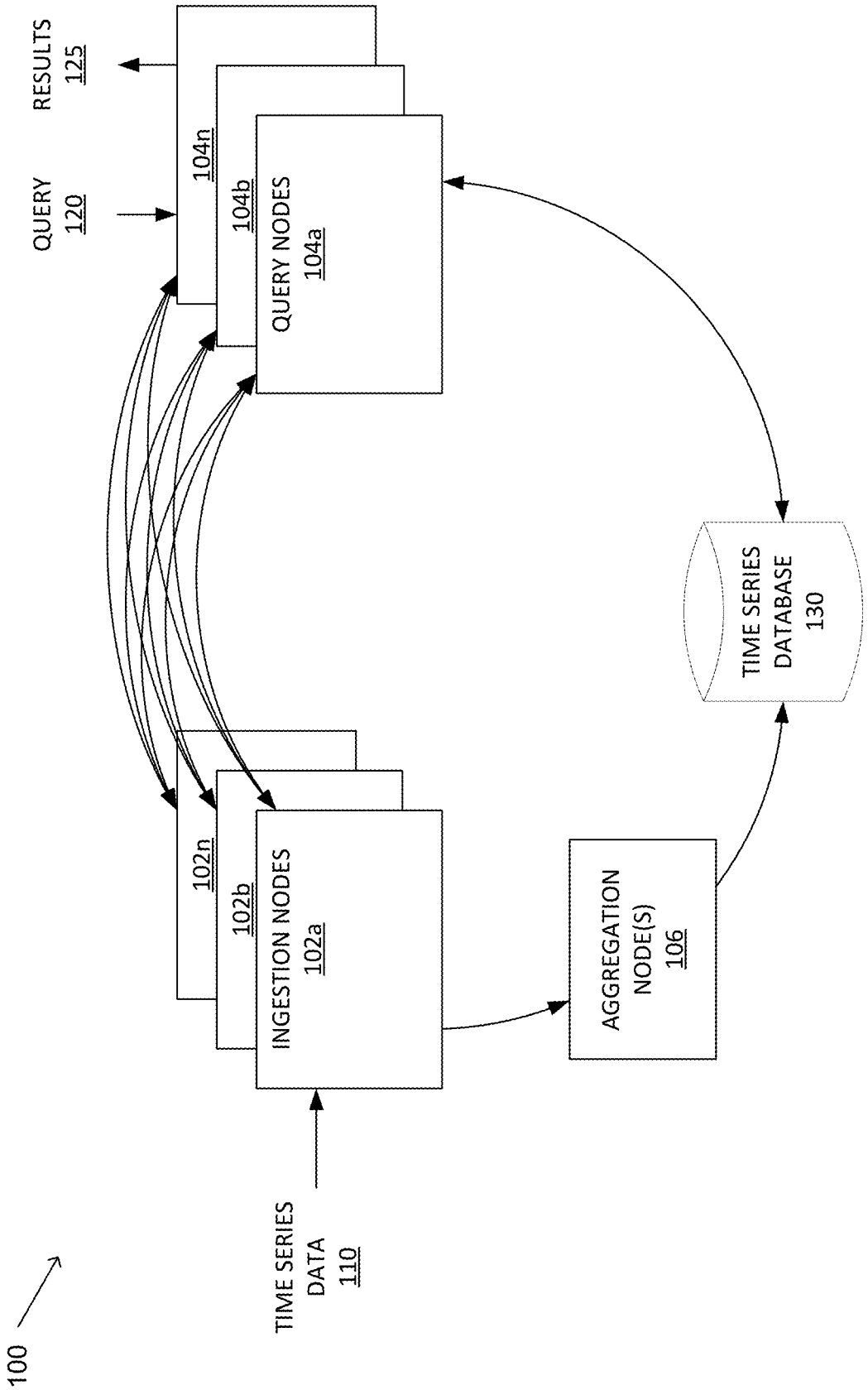

100
TIME SERIES DATA 110
INGESTION NODES 102a
102b
102n
AGGREGATION NODE(S) 106
TIME SERIES DATABASE 130
QUERY NODES 104a
104b
104n
QUERY 120
RESULTS 125

FIG. 1

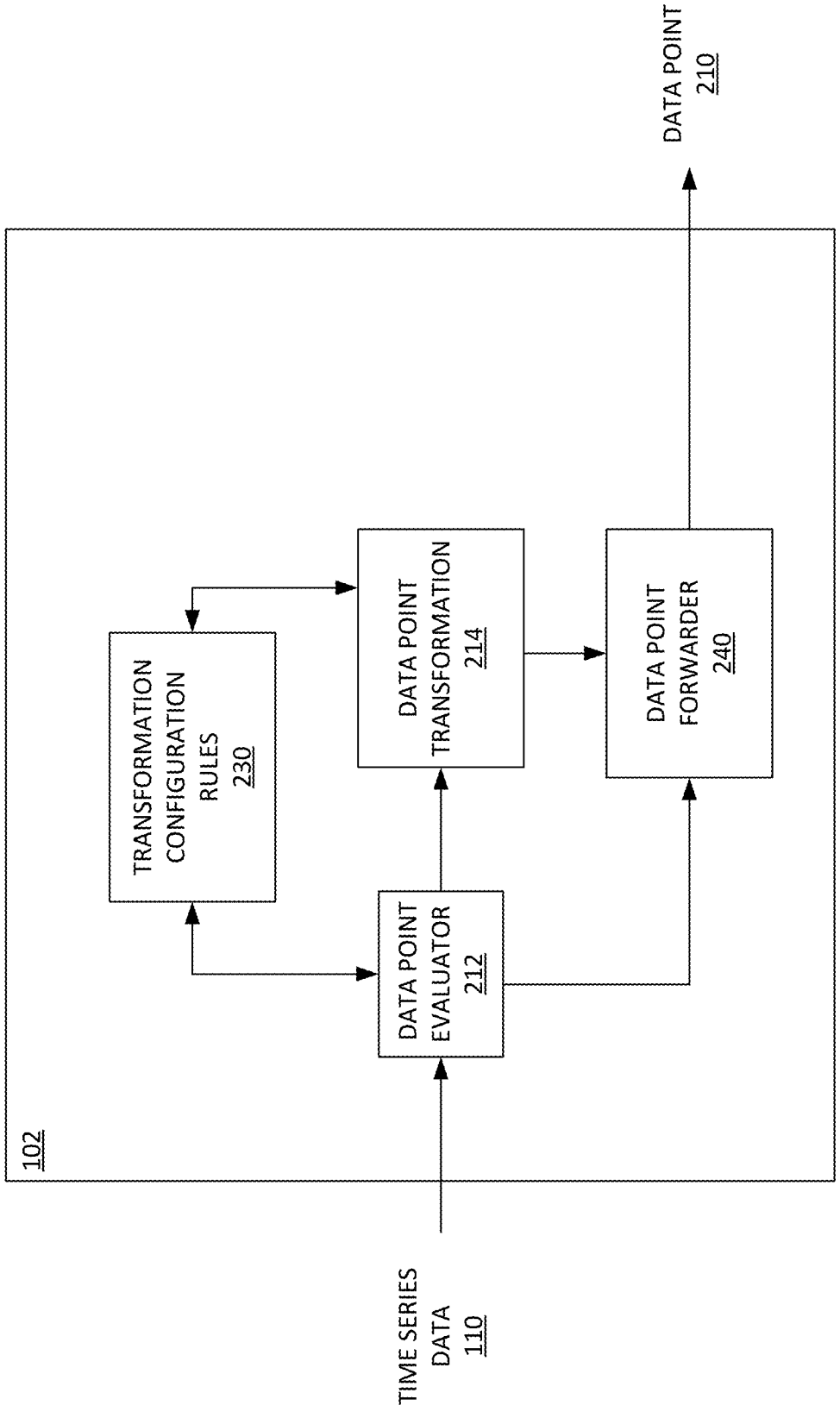

TIME SERIES DATA 110
102
DATA POINT EVALUATOR 212
TRANSFORMATION CONFIGURATION RULES 230
DATA POINT TRANSFORMATION 214
DATA POINT FORWARDER 240
DATA POINT 210

FIG. 2A

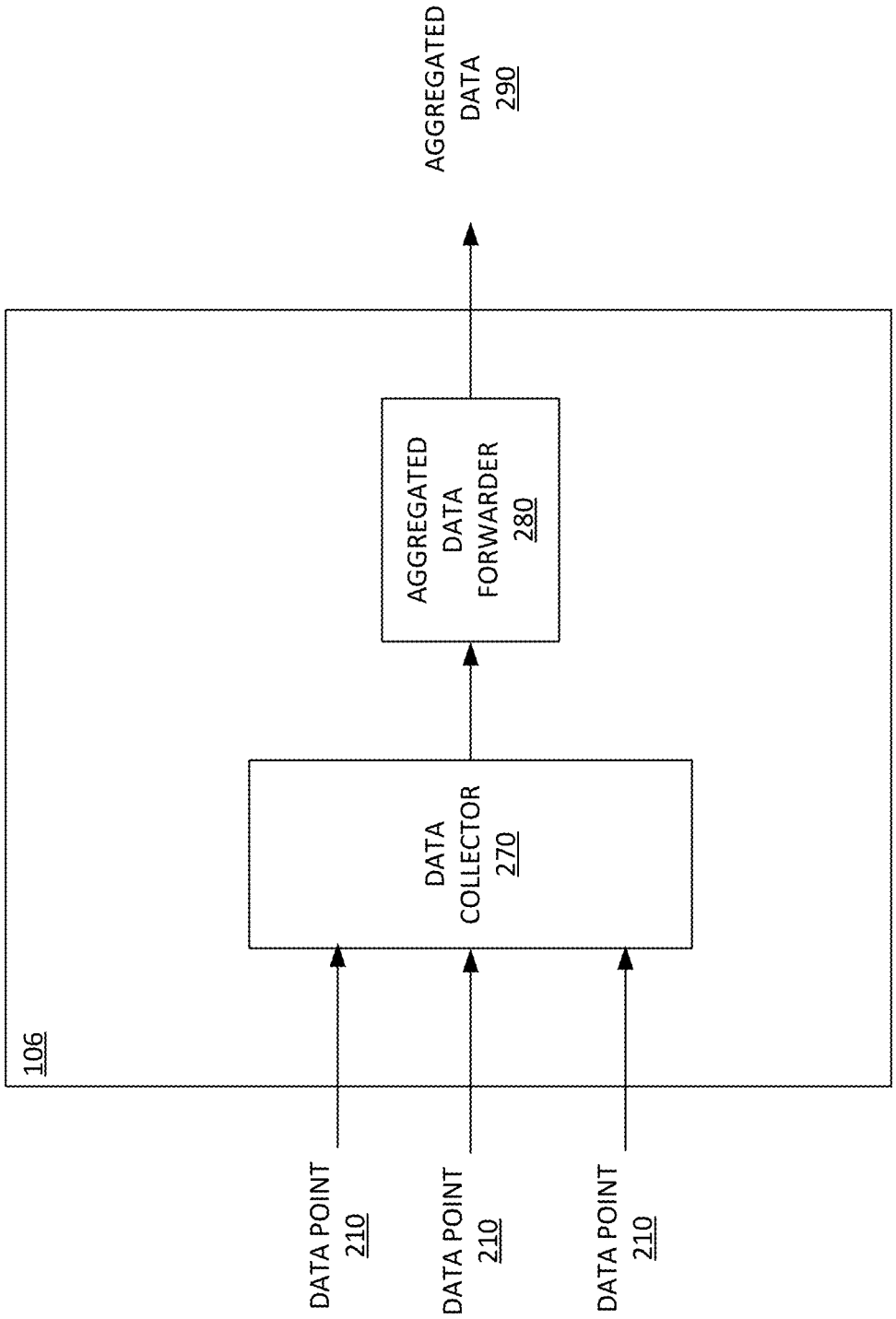

AGGREGATED DATA 290
AGGREGATED DATA FORWARDER 280
DATA COLLECTOR 270
106
DATA POINT 210
DATA POINT 210
DATA POINT 210

FIG. 2B

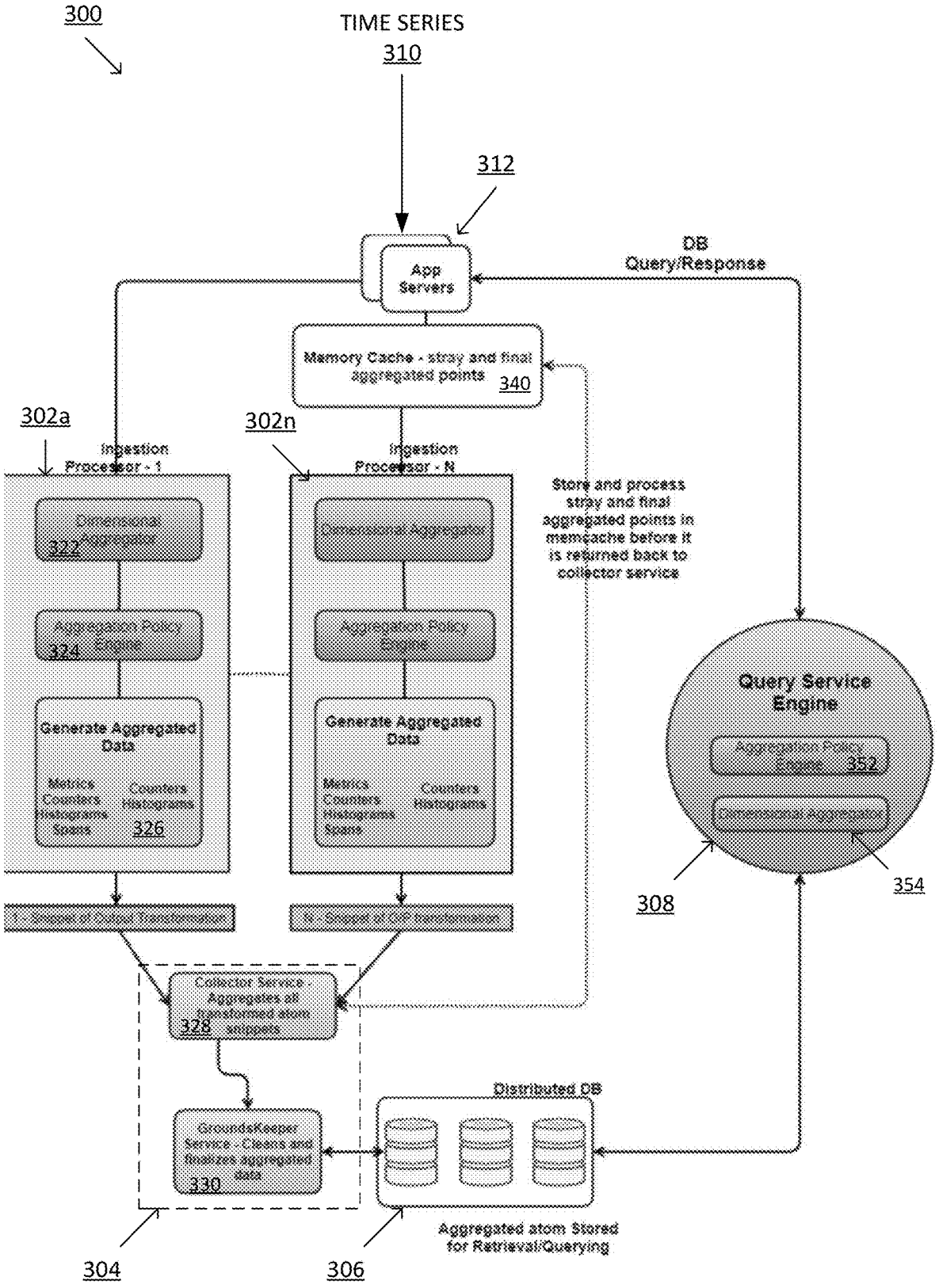

300
TIME SERIES
310
312
App Servers
DB Query/Response
Memory Cache - stray and final aggregated points
340
302a
302n
Ingestion Processor - 1
Ingestion Processor - N
Store and process stray and final aggregated points in memcache before it is returned back to collector service
Generate Aggregated Data
Metrics
Counters
Histograms
Spans
326
Query Service Engine
352
354
308
Collector Service - Aggregates all transformed atom streams
328
GroundsKeeper Service - Cleans and finalizes aggregated data
330
Distributed DB
Aggregated atom Stored for Retrieval/Querying
304
306

FIG. 3

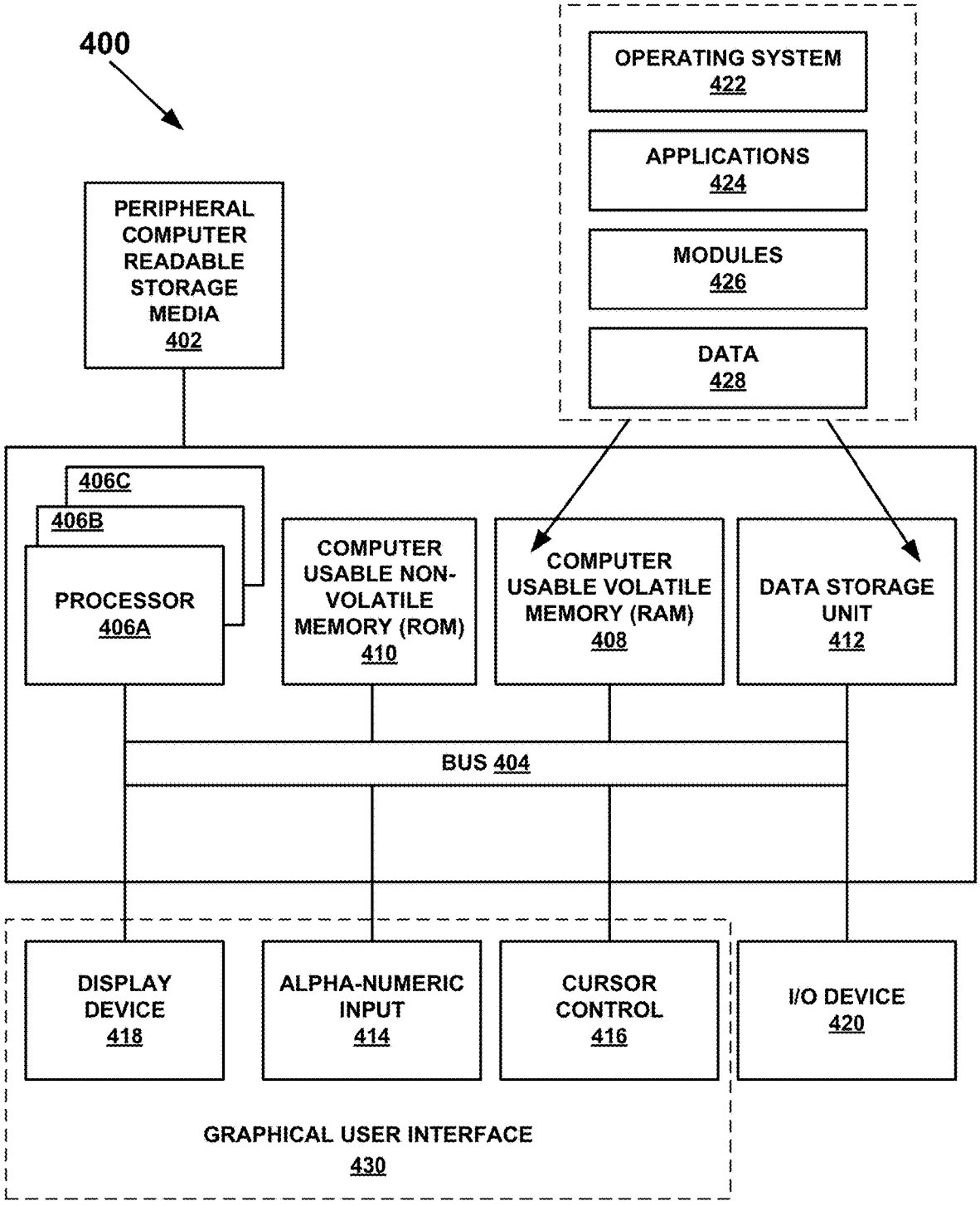

400
OPERATING SYSTEM
422
APPLICATIONS
424
MODULES
426
DATA
428
PERIPHERAL COMPUTER READABLE STORAGE MEDIA
402
406C
406B
PROCESSOR
406A
COMPUTER USABLE NON-VOLATILE MEMORY (ROM)
410
COMPUTER USABLE VOLATILE MEMORY (RAM)
408
DATA STORAGE UNIT
412
BUS 404
DISPLAY DEVICE
418
ALPHA-NUMERIC INPUT
414
CURSOR CONTROL
416
I/O DEVICE
420
GRAPHICAL USER INTERFACE
430

FIG. 4

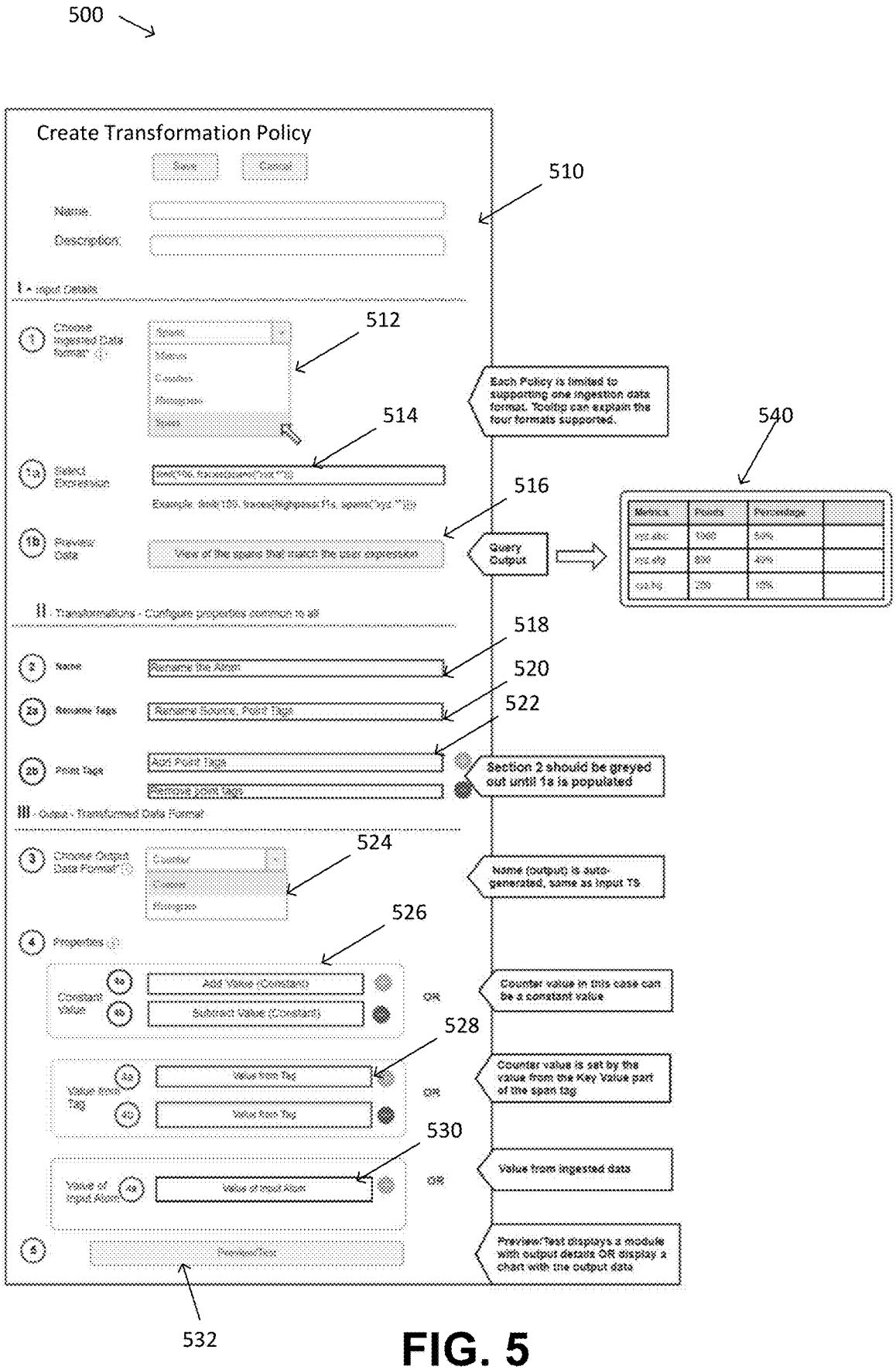

500
Create Transformation Policy
510
512
514
516
518
520
522
524
526
528
530
532
540
Each Policy is limited to supporting one ingestion data format. Tooltip can explain the four formats supported.
Query Output
Section 2 should be greyed out until 1a is populated
Name (output) is auto-generated, same as input TS
Counter value in this case can be a constant value
Counter value is set by the value from the Key Value part of the span tag
Value from ingested data
Preview/Test displays a module with output details OR display a chart with the output data

FIG. 5

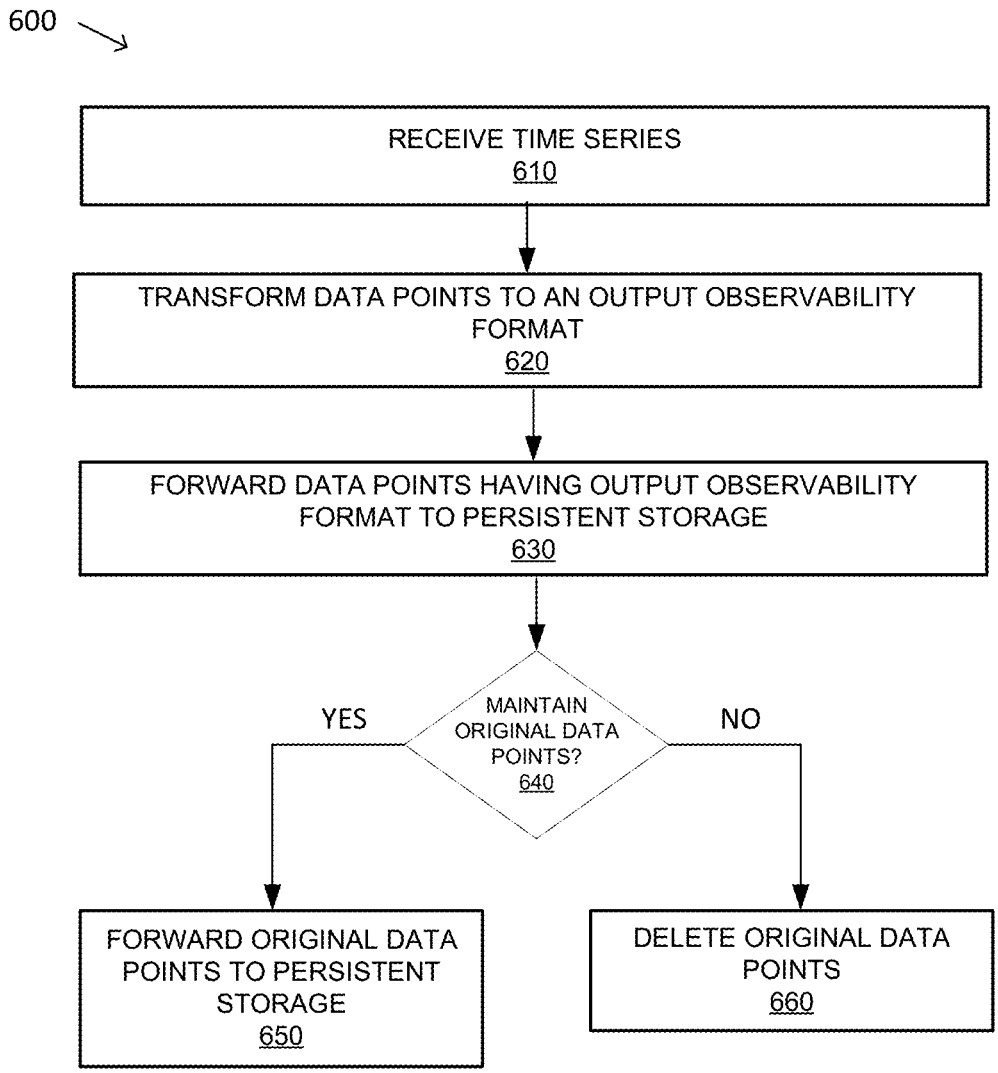

600
RECEIVE TIME SERIES
610
TRANSFORM DATA POINTS TO AN OUTPUT OBSERVABILITY FORMAT
620
FORWARD DATA POINTS HAVING OUTPUT OBSERVABILITY FORMAT TO PERSISTENT STORAGE
630
MAINTAIN ORIGINAL DATA POINTS?
640
YES
NO
FORWARD ORIGINAL DATA POINTS TO PERSISTENT STORAGE
650
DELETE ORIGINAL DATA POINTS
660

FIG. 6

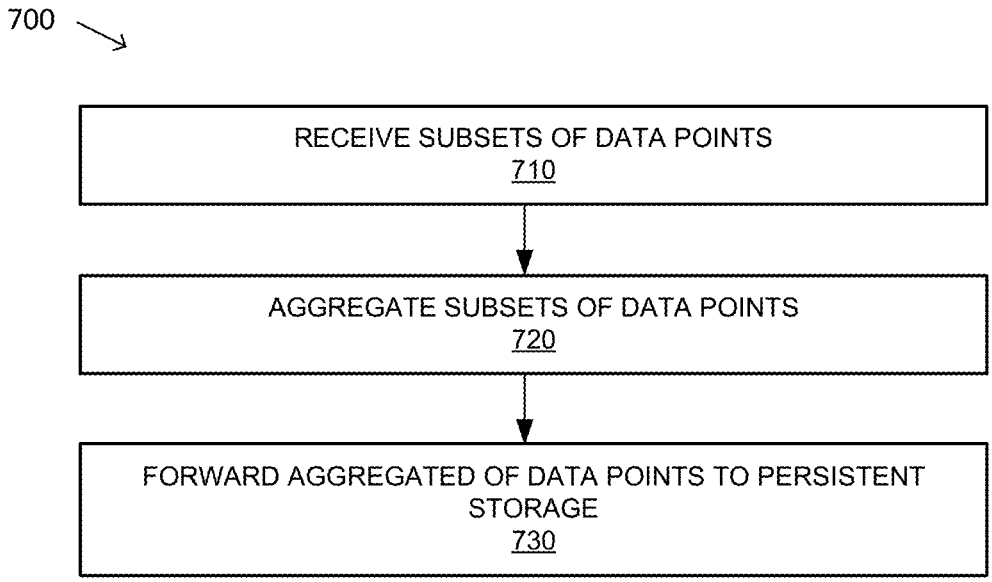

700
RECEIVE SUBSETS OF DATA POINTS
710
AGGREGATE SUBSETS OF DATA POINTS
720
FORWARD AGGREGATED OF DATA POINTS TO PERSISTENT STORAGE
730

FIG. 7

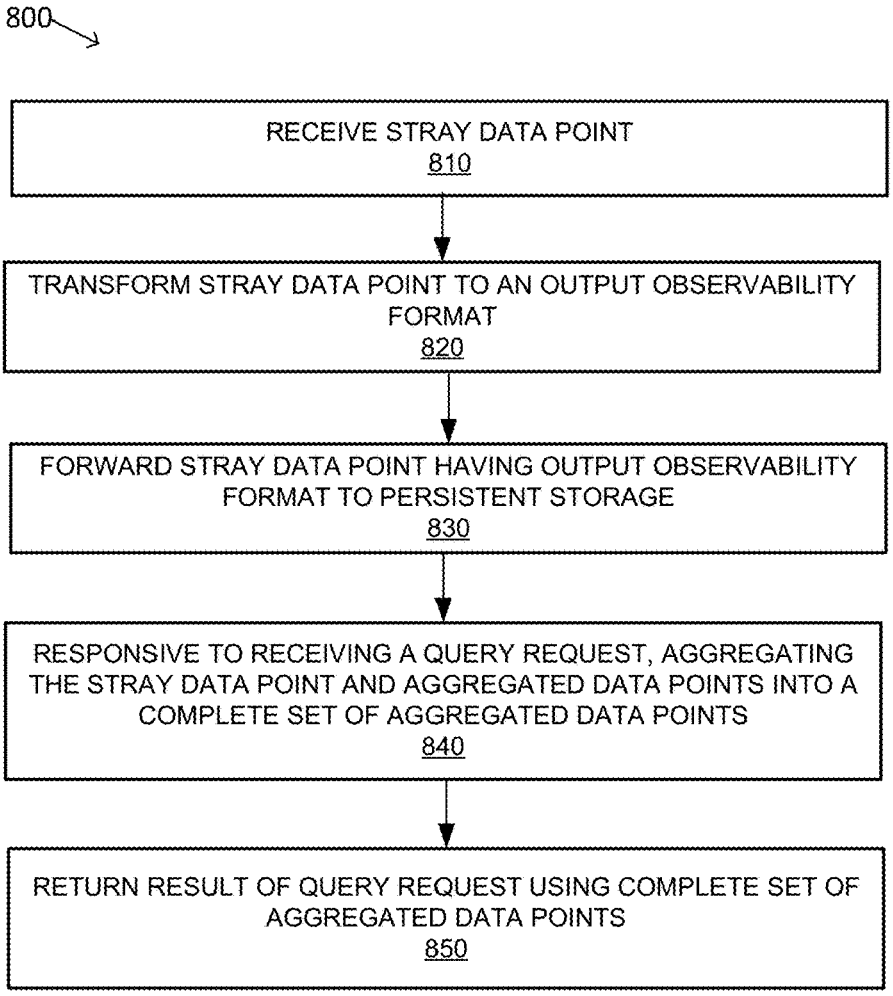

800
RECEIVE STRAY DATA POINT
810
TRANSFORM STRAY DATA POINT TO AN OUTPUT OBSERVABILITY FORMAT
820
FORWARD STRAY DATA POINT HAVING OUTPUT OBSERVABILITY FORMAT TO PERSISTENT STORAGE
830
RESPONSIVE TO RECEIVING A QUERY REQUEST, AGGREGATING THE STRAY DATA POINT AND AGGREGATED DATA POINTS INTO A COMPLETE SET OF AGGREGATED DATA POINTS
840
RETURN RESULT OF QUERY REQUEST USING COMPLETE SET OF AGGREGATED DATA POINTS
850

FIG. 8

AUTOMATIC TRANSFORMATION OF TIME SERIES DATA AT INGESTION

BACKGROUND

[0001] Management, monitoring, and troubleshooting in dynamic environments, both cloud-based and on-premises products, is increasingly important as the popularity of such products continues to grow. As the quantities of time-sensitive data grow, conventional techniques are increasingly deficient in the management of these applications. Conventional techniques, such as relational databases, have difficulty managing large quantities of data and have limited scalability. Moreover, as monitoring analytics of these large quantities of data often have real-time requirements, the deficiencies of reliance on relational databases become more pronounced.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] The accompanying drawings, which are incorporated in and form a part of this specification, illustrate various embodiments and, together with the Description of Embodiments, serve to explain principles discussed below. The drawings referred to in this brief description of the drawings should not be understood as being drawn to scale unless specifically noted.

[0003] FIG. **1** is a block diagram illustrating a time series data monitoring system for automatic transformation of time series data at ingestion, in accordance with embodiments.

[0004] FIG. **2**A is a block diagram illustrating an example ingestion node for automatic transformation of time series data at ingestion, in accordance with embodiments.

[0005] FIG. **2**B is a block diagram illustrating an example aggregation node of a system for automatic transformation of time series data at ingestion, in accordance with embodiments.

[0006] FIG. **3** is a block diagram illustrating an example time series data monitoring system for automatic transformation of time series data at ingestion, in accordance with embodiments.

[0007] FIG. **4** is a block diagram of an example computer system upon which embodiments of the present invention can be implemented.

[0008] FIG. **5** is an example graphical user interface for controlling configuration rules of a system for automatic transformation of time series data at ingestion.

[0009] FIG. **6** depicts a flow diagram for automatic transformation of time series data at ingestion, according to an embodiment.

[0010] FIG. **7** depicts a flow diagram for aggregating data in a system for automatic transformation of time series data at ingestion, according to an embodiment.

[0011] FIG. **8** depicts a flow diagram for automatic transformation a stray data point of time series data at ingestion, according to an embodiment.

DETAILED DESCRIPTION OF CERTAIN EMBODIMENTS

[0012] Reference will now be made in detail to various embodiments of the subject matter, examples of which are illustrated in the accompanying drawings. While various embodiments are discussed herein, it will be understood that they are not intended to limit to these embodiments. On the contrary, the presented embodiments are intended to cover alternatives, modifications and equivalents, which may be included within the spirit and scope the various embodiments as defined by the appended claims. Furthermore, in this Description of Embodiments, numerous specific details are set forth in order to provide a thorough understanding of embodiments of the present subject matter. However, embodiments may be practiced without these specific details. In other instances, well known methods, procedures, components, and circuits have not been described in detail as not to unnecessarily obscure aspects of the described embodiments.

[0013] Some portions of the detailed descriptions which follow are presented in terms of procedures, logic blocks, processing and other symbolic representations of operations on data bits within a computer memory. These descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. In the present application, a procedure, logic block, process, or the like, is conceived to be one or more self-consistent procedures or instructions leading to a desired result. The procedures are those requiring physical manipulations of physical quantities. Usually, although not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated in an electronic device.

[0014] It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussions, it is appreciated that throughout the description of embodiments, discussions utilizing terms such as "receiving," "transforming," "storing," "forwarding," "deleting," "aggregating," "returning," or the like, refer to the actions and processes of an electronic computing device or system such as: a host processor, a processor, a memory, a cloud-computing environment, a hyper-converged appliance, a software defined network (SDN) manager, a system manager, a virtualization management server or a virtual machine (VM), among others, of a virtualization infrastructure or a computer system of a distributed computing system, or the like, or a combination thereof. The electronic device manipulates and transforms data represented as physical (electronic and/or magnetic) quantities within the electronic device's registers and memories into other data similarly represented as physical quantities within the electronic device's memories or registers or other such information storage, transmission, processing, or display components.

[0015] Embodiments described herein may be discussed in the general context of processor-executable instructions residing on some form of non-transitory processor-readable medium, such as program modules, executed by one or more computers or other devices. Generally, program modules include routines, programs, objects, components, data structures, etc., that perform particular tasks or implement particular abstract data types. The functionality of the program modules may be combined or distributed as desired in various embodiments.

[0016] In the figures, a single block may be described as performing a function or functions; however, in actual practice, the function or functions performed by that block may be performed in a single component or across multiple components, and/or may be performed using hardware,

using software, or using a combination of hardware and software. To clearly illustrate this interchangeability of hardware and software, various illustrative components, blocks, modules, circuits, and steps have been described generally in terms of their functionality. Whether such functionality is implemented as hardware or software depends upon the particular application and design constraints imposed on the overall system. Skilled artisans may implement the described functionality in varying ways for each particular application, but such implementation decisions should not be interpreted as causing a departure from the scope of the present disclosure. Also, the example mobile electronic device described herein may include components other than those shown, including well-known components.

[0017] The techniques described herein may be implemented in hardware, software, firmware, or any combination thereof, unless specifically described as being implemented in a specific manner. Any features described as modules or components may also be implemented together in an integrated logic device or separately as discrete but interoperable logic devices. If implemented in software, the techniques may be realized at least in part by a non-transitory processor-readable storage medium comprising instructions that, when executed, perform one or more of the methods described herein. The non-transitory processor-readable data storage medium may form part of a computer program product, which may include packaging materials.

[0018] The non-transitory processor-readable storage medium may comprise random access memory (RAM) such as synchronous dynamic random access memory (SDRAM), read only memory (ROM), non-volatile random access memory (NVRAM), electrically erasable programmable read-only memory (EEPROM), FLASH memory, other known storage media, and the like. The techniques additionally, or alternatively, may be realized at least in part by a processor-readable communication medium that carries or communicates code in the form of instructions or data structures and that can be accessed, read, and/or executed by a computer or other processor.

[0019] The various illustrative logical blocks, modules, circuits and instructions described in connection with the embodiments disclosed herein may be executed by one or more processors, such as one or more motion processing units (MPUs), sensor processing units (SPUs), host processor(s) or core(s) thereof, digital signal processors (DSPs), general purpose microprocessors, application specific integrated circuits (ASICs), application specific instruction set processors (ASIPs), field programmable gate arrays (FPGAs), or other equivalent integrated or discrete logic circuitry. The term "processor," as used herein may refer to any of the foregoing structures or any other structure suitable for implementation of the techniques described herein. In addition, in some aspects, the functionality described herein may be provided within dedicated software modules or hardware modules configured as described herein. Also, the techniques could be fully implemented in one or more circuits or logic elements. A general purpose processor may be a microprocessor, but in the alternative, the processor may be any conventional processor, controller, microcontroller, or state machine. A processor may also be implemented as a combination of computing devices, e.g., a combination of an SPU/MPU and a microprocessor, a plurality of microprocessors, one or more microprocessors in conjunction with an SPU core, MPU core, or any other such configuration.

Overview of Discussion

[0020] Example embodiments described herein improve the performance of computer systems by transforming time series data at ingestion, rather than at query, and storing the transformed data in a persistent storage device. In accordance with various embodiments, a computer-implemented method for automatic transformation of time series data at ingestion is described. Time series data comprising data points is received at at least one ingestion node of a time series data monitoring system, wherein the data points have an input observability format. At the at least one ingestion node, the data points the data points are transformed from the input observability format to an output observability format according to configuration rules of the time series data monitoring system. The data points having the output observability format are forwarded from the at least one ingestion node to a persistent storage device.

[0021] In some embodiments, the configuration rules of the time series data monitoring system define operations for the transforming the data points from the input observability format to the output observability format. In some embodiments, the configuration rules identify input time series data necessitating transformation to the output observability format. In some embodiments, the input observability format is one of a metric, a counter, a histogram, and a span. In some embodiments, the output observability format is one of a counter and a histogram.

[0022] In one embodiment, the data points having the input observability format are forwarded from the at least one ingestion node to the persistent storage device. In another embodiment, the data points comprising the input observability format are deleted subsequent transformation to the output observability format.

[0023] In some embodiments, subsets of data points having the output observability format are received from a plurality of ingestion nodes at an intermediate aggregation node between the plurality of ingestion nodes and the persistent storage device. The subsets of data points having the output observability format from the plurality of ingestion nodes are aggregated into aggregated data points having the output observability format. In some embodiments, the aggregated data points having the output observability format are forwarded from the intermediate aggregation node to the persistent storage device.

[0024] In some embodiments, a stray data point of the time series data having the input observability format is received at the at least one ingestion node, the stray data point received subsequent the forwarding of the data points having the output observability format to the persistent storage device. The stray data point is transformed at the at least one ingestion node from the input observability format to the output observability format according to the configuration rules of the time series data monitoring system. The stray data point having the output observability format is forwarded from the at least one ingestion node to the persistent storage device. In some embodiments, responsive to receiving a query request associated with the data points having the output observability format and the stray data point having the output observability format, the data points having the output observability format and the stray data point having the output observability format are aggregated

into a complete set of aggregated data points having the output observability format. A result to the query request is returned using the complete set of aggregated data points.

[0025] Time series data can provide powerful insights into the performance of a system. The monitoring and analysis of time series data can provide large amounts of data for analysis. Due to volume of time series data typically received, as well as the frequency of receipt of the time series data, analysis of the data can be challenging. For instance, query processing may be time and processing intensive, as there are often data transformations that are required in order to respond to the query. Embodiments described herein provide for improved handling of query requests by transforming time series data from input observability atoms to output observability atoms such that a transformation is not necessary at query time. Moreover, in some embodiments, the input time series data can be discarded, allowing for improved memory management policies by only keeping the data that is needed for query processing in persistent storage.

[0026] Embodiments described herein provide users with the ability to define policies or rules that can transform time-series data ingested into a time series data monitoring system at the time of ingestion to an aggregated form of the same time-series data as ingested format or transform the data and store it even as a different time-series data format, also referred to herein as an "observability atom." For example, time series data having a histogram observability atom which can be transformed to a counter observability atom at ingestion. The time series data is then stored in persistent storage, e.g., a database, as the counter observability atom. In some embodiments, the transformation to a new observability atom at ingestion is performed in real-time. Embodiments described herein provide for transformation from one of four input observability atoms (e.g., spans, metrics, histograms, and counters) to one of two output observability atoms (e.g., counters and histograms).

[0027] As presented above, time series data monitoring systems typically process very large amounts of data, such that transformation of data to a different format or observability atom can be time-consuming and processing intensive. The efficient handling of data conversions can markedly improve performance of query processing. For instance, performing data transformation at the time of ingestion can improve query processing, by providing the data in a desired observability atom as the data is stored in the persistent storage, such that at query time no transformation of data is necessary,

[0028] The described embodiments speed up query processing and improve memory management, thereby improving the performance of the overall system. Hence, the embodiments of the present invention greatly extend beyond conventional methods of handling index updates of a time series data monitoring system. Moreover, embodiments of the present invention amount to significantly more than merely using a computer to perform the index updates. Instead, embodiments of the present invention specifically recite a novel process, rooted in computer technology, for automatic transformation of time series data at ingestion, to overcome a problem specifically arising in the realm of monitoring time series data and processing index updates on time series data within computer systems.

Example System for Managing Time Series Data

[0029] FIG. **1** is a block diagram illustrating an embodiment of a system **100** for automatic transformation of time series data at ingestion, according to embodiments. System **100** is a distributed system including multiple ingestion nodes **102***a* through **102***n* (collectively referred to herein as ingestion nodes **102**) and multiple query nodes **104***a* through **104***n* (collectively referred to herein as query nodes **104**). Time series **110** is received at ingestion nodes **102** and stored within time series database **130**. Query nodes **104** receive at least one query **120** for querying against time series database **130**. Results **125** of query **120** are returned upon execution of query **120**.

[0030] It should be appreciated that system **100** can include any number of ingestion nodes **102** and multiple query nodes **104**. Ingestion nodes **102** and query nodes **104** can be distributed over a network of computing devices in many different configurations. For example, the respective ingestion nodes **102** and query nodes **104** can be implemented where individual nodes independently operate and perform separate ingestion or query operations. In some embodiments, multiple nodes may operate on a particular computing device (e.g., via virtualization), while performing independently of other nodes on the computing device. In other embodiment, many copies of the service (e.g., ingestion or query) are distributed across multiple nodes (e.g., for purposes of reliability and scalability).

[0031] Time series data **110** is received at at least one ingestion node **102***a* through **102***n*. In some embodiments, time series data includes a numerical measurement of a system or activity that can be collected and stored as a metric (also referred to as a "stream"). For example, one type of metric is a CPU load measured over time. Other examples include, service uptime, memory usage, etc. It should be appreciated that metrics can be collected for any type of measurable performance of a system or activity. Operations can be performed on data points in a stream. In some instances, the operations can be performed in real time as data points are received. In other instances, the operations can be performed on historical data. Metrics analysis include a variety of use cases including online services (e.g., access to applications), software development, energy, Internet of Things (IoT), financial services (e.g., payment processing), healthcare, manufacturing, retail, operations management, and the like. It should be appreciated that the preceding examples are non-limiting, and that metrics analysis can be utilized in many different types of use cases and applications.

[0032] In accordance with some embodiments, a data point in a stream (e.g., in a metric) includes a name, a source, a value, and a time stamp. Optionally, a data point can include one or more tags (e.g., point tags). For example, a data point for a metric may include:

[0033] A name—the name of the metric (e.g., CPU_idle, service.uptime)

[0034] A source—the name of an application, host, container, instance, or other entity generating the metric (e.g., web_server_1, app1, app2)

[0035] A value—the value of the metric (e.g., 99% idle, 1000, 2000)

[0036] A timestamp—the timestamp of the metric (e.g., 1418436586000)

[0037] One or more point tags (optional)—custom metadata associated with the metric (e.g., location=las_vegas, environment=prod)

[0038] Ingestion nodes **102** are configured to process received data points of time series data **110** for persistence and indexing. In some embodiments, ingestion nodes **102** forward the data points of time series data **110** to time series database **130** for storage. In some embodiments, the data points of time series data **110** are transmitted to an intermediate buffer for handling the storage of the data points at time series database **130**. In one embodiment, time series database **130** can store and output time series data, e.g., TS1, TS2, TS3, etc. The data can include times series data, which may be discrete or continuous. For example, the data can include live data fed to a discrete stream, e.g., for a standing query. Continuous sources can include analog output representing a value as a function of time. With respect to processing operations, continuous data may be time sensitive, e.g., reacting to a declared time at which a unit of stream processing is attempted, or a constant, e.g., a 10V signal. Discrete streams can be provided to the processing operations in timestamp order. It should be appreciated that the time series data may be queried in real-time (e.g., by accessing the live data stream) or offline processing (e.g., by accessing the stored time series data).

[0039] In accordance with various embodiments, received data points of time series data **110** also have an associated input observability format, also referred to herein as "observability atoms." In some embodiments, the configuration rules of the time series data monitoring system define operations for the transforming the data points from the input observability atom to the output observability atom. In some embodiments, the configuration rules identify input time series data necessitating transformation to the output observability atom. In some embodiments, the input observability atom is one of a metric, a counter, a histogram, and a span. In some embodiments, wherein the output observability atom is one of a counter and a histogram.

[0040] FIG. **2**A is a block diagram illustrating an example ingestion node **102** (e.g., one of ingestion nodes **102***a* through **102***n* of FIG. **1**) for automatic transformation of time series data **110** at ingestion, in accordance with embodiments. In one embodiment, ingestion node **102** receives time series data **110** (e.g., as data points), evaluates whether data points of time series data **110** requires transformation from an input observability atom to an output observability atom, and performs the transformation when necessary. Ingestion node **102** includes data point evaluator **212**, data point transformation **214**, transformation configuration rules **230**, and data point forwarder **240**. It should be appreciated that ingestion node **102** is one node of a plurality of ingestion nodes of a distributed system for managing time series data (e.g., system **100**).

[0041] In the example shown in FIG. **2**, time series data **110** including data points is received. In one embodiment, time series data **110** including data points is received from an application or system. Time series data **110** is received at data point evaluator **212**. Data point evaluator **212** is configured to evaluate each data point according to transformation configuration rules **230** and determine whether a transformation of the data point from an input observability atom to an output observability atom is to be performed according to transformation configuration rules **230**. For example, configuration rules **230** may indicate that time series data **110** having a particular point tag or name is to be transformed from the input observability atom to a particular output observability atom.

[0042] In one embodiment, transformation configuration rules **230** include an indication of the input observability atom to be transformed. In accordance various embodiments, the input observability format is one of a metric, a counter, a histogram, and a span. Transformation configuration rules **230** also include an expression to select the ingested data points of time series **110** to be transformed, e.g., limit(100, traces(spans("xyz.*"))). Data point evaluator **212** scans the data points of time series **110** to identify data points that satisfy the expression, and then forwards the data points to data point transformation **214** to execute a transformation from the input observability atom to an output observability atom.

[0043] Responsive to determining that a data point does not require transformation to a different observability atom according to transformation configuration rules **230**, data point evaluator **212** forwards the data point **210** to data point forwarder **240** for ultimate forwarding to persistent storage. Data point forwarder **240** is configured to forward the data point **210** to persistent storage (e.g., time series database **130** of FIG. **1**).

[0044] Responsive to determining that a data point does require transformation to a different observability atom according to transformation configuration rules **230**, data point evaluator **212** forwards the data point to data point transformation **214**. Data point transformation **214** is configured to transform data points from an input observability atom to an output observability atom, according to transformation configuration rules **230**.

[0045] Data point transformation **214** receives the data points to be transformed, where each data point has a name, a source identifier, and one or more point tags (e.g., a set of point tags). Transformation configuration rules **230** allow for the configuration of a common set of transformation to the input data points having an input observability atom. In some embodiments, transformation configuration rules **230** have a priority order. Upon configuring the transformation configuration rules **230** for a transformation, the priority order can be set depending on which input observability atom will be transformed. Examples of the common set of transformation configuration rules **230** include, without limitation:

[0046] Rename the data point;

[0047] Rename a dimension of the data point (e.g., source, point tag);

[0048] Add a point tag;

[0049] Remove all point tags except listed point tags;

[0050] Drop the data point if the point tag is missing; and

[0051] Drop the data point if metric name matches

[0052] In accordance various embodiments, the output observability format is one of a counter and a histogram. The following are example operations describing the transformation from one of a metric, a counter, a histogram, and a span observability atom to one of a counter and a histogram observability atom.

[0053] In one embodiment, the transformation is from a metric observability atom to a counter observability atom. In one example transformation, the value of the counter can be set through four options. In the first option, the value of the metric is added as the delta of the counter. In the second

option, a constant value is added regardless of what the value of the underlying metric is (e.g., the value of the metric is ignored but this value is added to or subtracted from the counter). In the third option, the value of a point tag is used as the counter increment. In the fourth option, numerical transformation is performed (e.g., using the metric and transforming a value of the metric, such as dividing the value by a fixed value, and used the transformed value as the value used by the counter).

[0054] In one embodiment, the transformation is from a span observability atom to a counter observability atom. In one example transformation, the duration of the span is set as the value of the counter. The counter value in this case can be a constant value, where the value part of the key-value pair of the span is the value used in the counter.

[0055] In one embodiment, the transformation is from a histogram observability atom to a counter observability atom. In one example transformation, a median of the histogram is determined and put it into the counter, where the median can be one of:

[0056] P99 percentile aggregation of the histogram;

[0057] Number of Centroids;

[0058] Sum of all the counts; and

[0059] Number of Observations in a histogram.

[0060] In one embodiment, the transformation is from a counter observability atom to a counter observability atom. In one example transformation, the value of the counter as set as one of three options: static value, a value of the data point (e.g., direct transfer), or a value from the point tag.

[0061] In some embodiments, the transformation is to a histogram observability atom, where a histogram uses a numerical value and can be a sampled value (e.g., latency, count, etc.) In one embodiment, the transformation is from a metric observability atom to a histogram observability atom. In one example transformation, the transformation includes using one of a static value, using a value of the data point (e.g., a direct transfer), or a value from a point tag.

[0062] In one embodiment, the transformation is from a metric observability atom to a histogram observability atom. In one example transformation, the transformation includes using one of a value of a span or a value from a key-value pair of the span.

[0063] In one embodiment, the transformation is from a metric observability atom to a histogram observability atom. In one example transformation, the transformation includes using one of a value of the metric or a value of the key-value pair of the metric.

[0064] In one embodiment, the transformation is from a metric observability atom to a histogram observability atom. In one example transformation, the transformation includes using one of three options: static value, a value of the data point (e.g., direct transfer), or a value from the point tag.

[0065] Upon completing a transformation to an output observability atom, data point transformation **214** forwards the data point **210** to data point forwarder **240** for ultimate forwarding to persistent storage. It should be appreciated that in accordance with some embodiments, data point forwarder **240** forwards data points **210** to an intermediate node (e.g., an aggregation node) en route to persistent storage. In some embodiments, as described above, there are multiple ingestion nodes **102**, where each ingestion node only receives a subset of a time series data **110** received at the time series data monitoring system. Data points **210**, both those that are transformed and those that are note transformed, can be forwarded to an aggregation node for aggregating subsets (e.g., snippets) of data points.

[0066] FIG. **2**B is a block diagram illustrating an example aggregation node **106** of a system for automatic transformation of time series data at ingestion, in accordance with embodiments. Aggregation node **106** includes data collector **270** for receiving and aggregating data points **210** into aggregated data **290**. The aggregated data **290** is then forwarded by aggregated data forwarder **280** to the next node in the system, e.g., a persistent storage node. In some embodiments, there are multiple layers of aggregation nodes **106**, such that a plurality of aggregation nodes **106** receive data points **210** of time series data from multiple ingestion nodes, and then forward the aggregated data **290** to another higher-level aggregation node **106**, which then aggregates the received aggregated data **290** and forwards aggregated data **290** to the persistent storage node. It should be appreciated that there can be any number of layers of aggregation nodes.

[0067] FIG. **3** is a block diagram illustrating an example time series data monitoring system **300** for automatic transformation of time series data at ingestion, in accordance with embodiments. System **300** is a distributed system including multiple ingestion nodes or processors **302***a* through **302***n* (collectively referred to herein as ingestion nodes or processors **302**), an aggregation node **304**, a distributed database **306**, and a query service engine **308**. Time series **310** is received at ingestion nodes **302**, in some embodiments via application servers **312**. Query service engine **308** may be implemented within and distributed over one or more query nodes (e.g., query nodes **104***a* through **104***n* of FIG. **1**).

[0068] It should be appreciated that system **300** can include any number of ingestion nodes **302** and multiple query nodes. Ingestion nodes **302** and the query nodes can be distributed over a network of computing devices in many different configurations. For example, the respective ingestion nodes **302** and query nodes can be implemented where individual nodes independently operate and perform separate ingestion or query operations. In some embodiments, multiple nodes may operate on a particular computing device (e.g., via virtualization), while performing independently of other nodes on the computing device. In other embodiment, many copies of the service (e.g., ingestion or query) are distributed across multiple nodes (e.g., for purposes of reliability and scalability).

[0069] Time series data **310** is received at at least one ingestion node **302***a* through **302***n*. In accordance with various embodiments, received data points of time series data **310** also have an associated input observability format, also referred to herein as "observability atoms." In some embodiments, a load balancer distributes time series **110** over ingestion node **302***a* through **302***n*, for purposes of handling the volume of time series **110** in real-time. Each data point of time series **110** is received and processed at an ingestion node **302** for purposes of determining whether the data point should be transformed into a different observability atom (e.g., as described in FIGS. **1**, **2**A, and **2**B, and for performing aggregation on the data points at dimensional aggregator **322** in accordance with an aggregation policy as defined by aggregation policy engine **324**. Dimensional aggregator **322** receives the data points having an input observability atom and performs transformation and/or aggregation in accordance with aggregation policy engine **324** to generate aggregated data **326**.

[0070] In some embodiments, the aggregation policy as defined by aggregation policy engine **324** are configuration rules that define operations for the transforming the data points from the input observability format to the output observability format (e.g., as described above at FIG. **2**A). In some embodiments, the configuration rules identify input time series data necessitating transformation to the output observability format. In some embodiments, the input observability format is one of a metric, a counter, a histogram, and a span. In some embodiments, the output observability format is one of a counter and a histogram.

[0071] Aggregated data **326** is output from ingestion node **302** as a subset (e.g., snippet) of the total aggregated data for system **300** and received at an intermediate aggregation node **304**. In one embodiment, aggregation node **304** includes collector service **328** for aggregating all the transformed data points and groundskeeper service **330** for cleaning up and finalizing the aggregated data for forwarding to distributed database **306** (e.g., persistent storage).

[0072] In some embodiments, a stray data point of the time series **110** having the input observability format is received at ingestion node **302** after the ingestion and transformation of time series **110** subject to the same transformation as indicated by the configuration rules (e.g., due to system latency or network issues). The stray data point is received after the rest of time series **110** is forwarded to and stored at database **306**. In these embodiments, the stray data point is transformed at the at least one ingestion node **302** from the input observability format to the output observability format according to the configuration rules of system **300**. The stray data point is forwarded from the ingestion node **302** to the database **306**. In some embodiments, responsive to receiving a query request from query service engine **308**, the query request associated with the time series **110**, the data points having the output observability format and the stray data point having the output observability format are aggregated into a complete set of aggregated data points having the output observability format by dimensional aggregator **354** which performs aggregation of time series **110** and the stray data point in accordance with aggregation policy engine **352** to generate complete aggregated data. In other embodiments, the stray data point and time series **110** are stored at a memory cache **340**. A result to the query request is returned using the complete set of aggregated data points.

[0073] Hence, the embodiments of the present invention greatly extend beyond conventional methods of handling query processing a time series data monitoring system. The described embodiments speed up query processing and improve memory management, thereby improving the performance of the overall system. Hence, the embodiments of the present invention greatly extend beyond conventional methods of handling index updates of a time series data monitoring system. Moreover, embodiments of the present invention amount to significantly more than merely using a computer to perform the index updates. Instead, embodiments of the present invention specifically recite a novel process, rooted in computer technology, for automatic transformation of time series data at ingestion, to overcome a problem specifically arising in the realm of monitoring time series data and processing index updates on time series data within computer systems.

[0074] FIG. **4** is a block diagram of an example computer system **400** upon which embodiments of the present invention can be implemented. FIG. **4** illustrates one example of a type of computer system **400** (e.g., a computer system) that can be used in accordance with or to implement various embodiments which are discussed herein.

[0075] It is appreciated that computer system **400** of FIG. **4** is only an example and that embodiments as described herein can operate on or within a number of different computer systems including, but not limited to, general purpose networked computer systems, embedded computer systems, mobile electronic devices, smart phones, server devices, client devices, various intermediate devices/nodes, standalone computer systems, media centers, handheld computer systems, multi-media devices, and the like. In some embodiments, computer system **400** of FIG. **4** is well adapted to having peripheral tangible computer-readable storage media **402** such as, for example, an electronic flash memory data storage device, a floppy disc, a compact disc, digital versatile disc, other disc based storage, universal serial bus "thumb" drive, removable memory card, and the like coupled thereto. The tangible computer-readable storage media is non-transitory in nature.

[0076] Computer system **400** of FIG. **4** includes an address/data bus **404** for communicating information, and a processor **406**A coupled with bus **404** for processing information and instructions. As depicted in FIG. **4**, computer system **400** is also well suited to a multi-processor environment in which a plurality of processors **406**A, **406**B, and **406**C are present. Conversely, computer system **400** is also well suited to having a single processor such as, for example, processor **406**A. Processors **406**A, **406**B, and **406**C may be any of various types of microprocessors. Computer system **400** also includes data storage features such as a computer usable volatile memory **408**, e.g., random access memory (RAM), coupled with bus **404** for storing information and instructions for processors **406**A, **406**B, and **406**C. Computer system **400** also includes computer usable non-volatile memory **410**, e.g., read only memory (ROM), coupled with bus **404** for storing static information and instructions for processors **406**A, **406**B, and **406**C. Also present in computer system **400** is a data storage unit **412** (e.g., a magnetic or optical disc and disc drive) coupled with bus **404** for storing information and instructions. Computer system **400** also includes an alphanumeric input device **414** including alphanumeric and function keys coupled with bus **404** for communicating information and command selections to processor **406**A or processors **406**A, **406**B, and **406**C. Computer system **400** also includes an cursor control device **416** coupled with bus **404** for communicating user input information and command selections to processor **406**A or processors **406**A, **406**B, and **406**C. In one embodiment, computer system **400** also includes a display device **418** coupled with bus **404** for displaying information.

[0077] Referring still to FIG. **4**, display device **418** of FIG. **4** may be a liquid crystal device (LCD), light emitting diode display (LED) device, cathode ray tube (CRT), plasma display device, a touch screen device, or other display device suitable for creating graphic images and alphanumeric characters recognizable to a user. Cursor control device **416** allows the computer user to dynamically signal the movement of a visible symbol (cursor) on a display screen of display device **418** and indicate user selections of selectable items displayed on display device **418**. Many implementations of cursor control device **416** are known in the art including a trackball, mouse, touch pad, touch screen, joystick or special keys on alphanumeric input device **414**

capable of signaling movement of a given direction or manner of displacement. Alternatively, it will be appreciated that a cursor can be directed and/or activated via input from alphanumeric input device **414** using special keys and key sequence commands. Computer system **400** is also well suited to having a cursor directed by other means such as, for example, voice commands. In various embodiments, alphanumeric input device **414**, cursor control device **416**, and display device **418**, or any combination thereof (e.g., user interface selection devices), may collectively operate to provide a graphical user interface (GUI) **430** under the direction of a processor (e.g., processor **406**A or processors **406**A, **406**B, and **406**C). GUI **430** allows user to interact with computer system **400** through graphical representations presented on display device **418** by interacting with alphanumeric input device **414** and/or cursor control device **416**.

[0078] Computer system **400** also includes an I/O device **420** for coupling computer system **400** with external entities. For example, in one embodiment, I/O device **420** is a modem for enabling wired or wireless communications between computer system **400** and an external network such as, but not limited to, the Internet. In one embodiment, I/O device **420** includes a transmitter. Computer system **400** may communicate with a network by transmitting data via I/O device **420**.

[0079] Referring still to FIG. **4**, various other components are depicted for computer system **400**. Specifically, when present, an operating system **422**, applications **424**, modules **426**, and data **428** are shown as typically residing in one or some combination of computer usable volatile memory **408** (e.g., RAM), computer usable non-volatile memory **410** (e.g., ROM), and data storage unit **412**. In some embodiments, all or portions of various embodiments described herein are stored, for example, as an application **424** and/or module **426** in memory locations within RAM **408**, computer-readable storage media within data storage unit **412**, peripheral computer-readable storage media **402**, and/or other tangible computer-readable storage media.

Example Graphical User Interface

[0080] FIG. **5** is an example graphical user interface **500** for receiving configuration rules of a system for automatic transformation of time series data at ingestion. At fields **510**, a name and description of the transformation policy can be entered. At drop-down menu **512**, a format of the ingested data can be selected. In one embodiment, the input observability format is one of a metric, a counter, a histogram, and a span. At field **514**, an expression can be entered for identifying data to be transformed. Selection of button **516** allows for the presentation of preview data **540** of input data that matches the user expression.

[0081] Fields **518**, **520**, and **522** allow for a user to configure common properties of the transformation of the data. At field **518**, the input observability atom can be renamed. At field **520**, tags (e.g., the source and point tags) of the input data can be renamed. At fields **522**, point tags can be added or removed from the input data.

[0082] At drop-down menu **524**, an output format of the transformed data can be selected. In one embodiment, the output observability format is one of a counter and a histogram. At field **526**, constant values can be added to or subtracted from the input data. At field **528**, values can be added to or removed from the tag. At field **530**, a value of the ingested data can be added. Selection of button **532** executes a preview or test of the transformation policy.

Example Methods of Operation

[0083] The following discussion sets forth in detail the operation of some example methods of operation of embodiments. With reference to FIGS. **6** through **8**, flow diagrams **600**, **700**, and **800** illustrate example procedures used by various embodiments. The flow diagrams **600**, **700**, and **800** include some procedures that, in various embodiments, are carried out by a processor under the control of computer-readable and computer-executable instructions. In this fashion, procedures described herein and in conjunction with the flow diagrams are, or may be, implemented using a computer, in various embodiments. The computer-readable and computer-executable instructions can reside in any tangible computer readable storage media. Some non-limiting examples of tangible computer readable storage media include random access memory, read only memory, magnetic disks, solid state drives/"disks," and optical disks, any or all of which may be employed with computer environments (e.g., computer system **400**). The computer-readable and computer-executable instructions, which reside on tangible computer readable storage media, are used to control or operate in conjunction with, for example, one or some combination of processors of the computer environments and/or virtualized environment. It is appreciated that the processor(s) may be physical or virtual or some combination (it should also be appreciated that a virtual processor is implemented on physical hardware). Although specific procedures are disclosed in the flow diagram, such procedures are examples. That is, embodiments are well suited to performing various other procedures or variations of the procedures recited in the flow diagram. Likewise, in some embodiments, the procedures in flow diagrams **600**, **700**, and **800** may be performed in an order different than presented and/or not all of the procedures described in flow diagrams **600**, **700**, and **800** may be performed. It is further appreciated that procedures described in flow diagrams **600**, **700**, and **800** may be implemented in hardware, or a combination of hardware with firmware and/or software provided by computer system **400**.

[0084] FIG. **6** depicts a flow diagram **600** for automatic transformation of time series data at ingestion, according to an embodiment. At procedure **610** of flow diagram **600**, time series data comprising data points is received at at least one ingestion node of a time series data monitoring system, wherein the data points have an input observability format. At procedure **620**, at the at least one ingestion node, the data points the data points are transformed from the input observability format to an output observability format according to configuration rules of the time series data monitoring system. In some embodiments, the configuration rules of the time series data monitoring system define operations for the transforming the data points from the input observability format to the output observability format. In some embodiments, the configuration rules identify input time series data necessitating transformation to the output observability format. In some embodiments, the input observability format is one of a metric, a counter, a histogram, and a span. In some embodiments, the output observability format is one of a counter and a histogram. At procedure **630**, the data points having the output observability format are forwarded from the at least one ingestion node to a persistent storage device.

[0085] In some embodiments, as shown at procedure **640**, it is determined whether to maintain the original data points having the input observability format. Provided it is determined to maintain the original data points having the input observability format, as shown at procedure **650**, the data points having the input observability format are forwarded from the at least one ingestion node to the persistent storage device. Provided it is determined not to maintain the original data points having the input observability format, as shown at procedure **660**, the data points comprising the input observability format are deleted subsequent transformation to the output observability format.

[0086] In some embodiments, there are one or more intermediate aggregation nodes between the ingestion nodes and the persistent storage. FIG. **7** depicts a flow diagram **700** for aggregating data in a system for automatic transformation of time series data at ingestion, according to an embodiment. At procedure **710** of flow diagram **700**, subsets of data points having the output observability format are received from a plurality of ingestion nodes at an intermediate aggregation node between the plurality of ingestion nodes and the persistent storage device. At procedure **720**, the subsets of data points having the output observability format from the plurality of ingestion nodes are aggregated into aggregated data points having the output observability format. In some embodiments, as show at procedure **730**, the aggregated data points having the output observability format are forwarded from the intermediate aggregation node to the persistent storage device.

[0087] FIG. **8** depicts a flow diagram **800** for automatic transformation a stray data point of time series data at ingestion, according to an embodiment. At procedure **810** of flow diagram **800**, a stray data point of the time series data having the input observability format is received at the at least one ingestion node, the stray data point received subsequent the forwarding of the data points having the output observability format to the persistent storage device. At procedure **820**, the stray data point is transformed at the at least one ingestion node from the input observability format to the output observability format according to the configuration rules of the time series data monitoring system. At procedure **830**, the stray data point having the output observability format is forwarded from the at least one ingestion node to the persistent storage device. In some embodiments, as shown at procedure **840**, responsive to receiving a query request associated with the data points having the output observability format and the stray data point having the output observability format, the data points having the output observability format and the stray data point having the output observability format are aggregated into a complete set of aggregated data points having the output observability format. At procedure **850**, a result to the query request is returned using the complete set of aggregated data points.

[0088] It is noted that any of the procedures, stated above, regarding the flow diagrams of FIGS. **6** through **8** may be implemented in hardware, or a combination of hardware with firmware and/or software. For example, any of the procedures are implemented by a processor(s) of a cloud environment and/or a computing environment.

[0089] One or more embodiments of the present invention may be implemented as one or more computer programs or as one or more computer program modules embodied in one or more computer readable media. The term computer readable medium refers to any data storage device that can store data which can thereafter be input to a computer system—computer readable media may be based on any existing or subsequently developed technology for embodying computer programs in a manner that enables them to be read by a computer. Examples of a computer readable medium include a hard drive, network attached storage (NAS), read-only memory, random-access memory (e.g., a flash memory device), a CD (Compact Discs)—CD-ROM, a CD-R, or a CD-RW, a DVD (Digital Versatile Disc), a magnetic tape, and other optical and non-optical data storage devices. The computer readable medium can also be distributed over a network coupled computer system so that the computer readable code is stored and executed in a distributed fashion.

[0090] Although one or more embodiments of the present invention have been described in some detail for clarity of understanding, it will be apparent that certain changes and modifications may be made within the scope of the claims. Accordingly, the described embodiments are to be considered as illustrative and not restrictive, and the scope of the claims is not to be limited to details given herein, but may be modified within the scope and equivalents of the claims. In the claims, elements and/or steps do not imply any particular order of operation, unless explicitly stated in the claims.

[0091] Many variations, modifications, additions, and improvements are possible, regardless the degree of virtualization. Plural instances may be provided for components, operations or structures described herein as a single instance. Finally, boundaries between various components, operations and data stores are somewhat arbitrary, and particular operations are illustrated in the context of specific illustrative configurations. Other allocations of functionality are envisioned and may fall within the scope of the invention(s). In general, structures and functionality presented as separate components in exemplary configurations may be implemented as a combined structure or component. Similarly, structures and functionality presented as a single component may be implemented as separate components. These and other variations, modifications, additions, and improvements may fall within the scope of the appended claims(s).

What is claimed is:

1. A computer-implemented method for automatic transformation of time series data at ingestion, the method comprising:

receiving time series data comprising data points at at least one ingestion node of a time series data monitoring system, wherein the data points have an input observability format;

transforming, at the at least one ingestion node, the data points from the input observability format to an output observability format according to configuration rules of the time series data monitoring system; and

forwarding, from the at least one ingestion node, the data points having the output observability format to a persistent storage device.

2. The method of claim **1**, further comprising:

forwarding, from the at least one ingestion node, the data points having the input observability format to the persistent storage device.

3. The method of claim 1, further comprising:

deleting the data points comprising the input observability format subsequent transformation to the output observability format.

4. The method of claim 1, further comprising:

receiving subsets of data points having the output observability format from a plurality of ingestion nodes at an intermediate aggregation node between the plurality of ingestion nodes and the persistent storage device; and

aggregating the subsets of data points having the output observability format from the plurality of ingestion nodes into aggregated data points having the output observability format.

5. The method of claim 4, further comprising:

forwarding, from the intermediate aggregation node, the aggregated data points having the output observability format to the persistent storage device.

6. The method of claim 1, further comprising:

receiving, at the at least one ingestion node, a stray data point of the time series data, the stray data point received subsequent the forwarding of the data points having the output observability format to the persistent storage device, wherein the stray data point has the input observability format;

transforming, at the at least one ingestion node, the stray data point from the input observability format to the output observability format according to the configuration rules of the time series data monitoring system; and

forwarding, from the at least one ingestion node, the stray data point having the output observability format to the persistent storage device.

7. The method of claim 6, further comprising:

responsive to receiving a query request associated with the data points having the output observability format and the stray data point having the output observability format, aggregating the data points having the output observability format and the stray data point having the output observability format into a complete set of aggregated data points having the output observability format; and

returning a result to the query request using the complete set of aggregated data points.

8. The method of claim 1, wherein the configuration rules of the time series data monitoring system define operations for the transforming the data points from the input observability format to the output observability format.

9. The method of claim 1, wherein the input observability format is one of a metric, a counter, a histogram, and a span.

10. The method of claim 1, wherein the output observability format is one of a counter and a histogram.

11. A non-transitory computer readable storage medium having computer readable program code stored thereon for causing a computer system to perform a method for automatic transformation of time series data at ingestion, the method comprising:

receiving time series data comprising data points at at least one ingestion node of a time series data monitoring system, wherein the data points have an input observability format;

transforming, at the at least one ingestion node, the data points from the input observability format to an output observability format according to configuration rules of the time series data monitoring system; and

forwarding, from the at least one ingestion node, the data points having the output observability format to a persistent storage device.

12. The non-transitory computer readable storage medium of claim 11, the method further comprising:

forwarding, from the at least one ingestion node, the data points having the input observability format to the persistent storage device.

13. The non-transitory computer readable storage medium of claim 11, the method further comprising:

deleting the data points comprising the input observability format subsequent transformation to the output observability format.

14. The non-transitory computer readable storage medium of claim 11, the method further comprising:

receiving subsets of data points having the output observability format from a plurality of ingestion nodes at an intermediate aggregation node between the plurality of ingestion nodes and the persistent storage device; and

aggregating the subsets of data points having the output observability format from the plurality of ingestion nodes into aggregated data points having the output observability format.

15. The non-transitory computer readable storage medium of claim 14, the method further comprising:

forwarding, from the intermediate aggregation node, the aggregated data points having the output observability format to the persistent storage device.

16. The non-transitory computer readable storage medium of claim 11, the method further comprising:

receiving, at the at least one ingestion node, a stray data point of the time series data, the stray data point received subsequent the forwarding of the data points having the output observability format to the persistent storage device, wherein the stray data point has the input observability format;

transforming, at the at least one ingestion node, the stray data point from the input observability format to the output observability format according to the configuration rules of the time series data monitoring system; and

forwarding, from the at least one ingestion node, the stray data point having the output observability format to the persistent storage device.

17. The non-transitory computer readable storage medium of claim 16, the method further comprising:

responsive to receiving a query request associated with the data points having the output observability format and the stray data point having the output observability format, aggregating the data points having the output observability format and the stray data point having the output observability format into a complete set of aggregated data points having the output observability format; and

returning a result to the query request using the complete set of aggregated data points.

18. The non-transitory computer readable storage medium of claim 11, wherein the configuration rules of the time series data monitoring system define operations for the transforming the data points from the input observability format to the output observability format.

19. The non-transitory computer readable storage medium of claim 11, wherein the input observability format is one of

a metric, a counter, a histogram, and a span and the output observability format is one of a counter and a histogram.

20. A time series data monitoring system for automatic transformation of time series data at ingestion, the time series data monitoring system comprising:

a persistent storage device;

a plurality of ingestion nodes, each node of the plurality of ingestion nodes comprising a data storage unit and a processor communicatively coupled with the data storage unit, wherein an ingestion node of the plurality of ingestion nodes is configured to:

receive time series data comprising data points, wherein the data points have an input observability format;

transform the data points from the input observability format to an output observability format according to configuration rules of the time series data monitoring system; and

forwarding the data points having the output observability format to the persistent storage device.

* * * * *