US 20200044927A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2020/0044927 A1**

Apostolopoulos et al. (43) **Pub. Date: Feb. 6, 2020**

(54) **BEHAVIORAL BASED DEVICE CLUSTERING SYSTEM AND METHOD**

(71) Applicant: **Splunk Inc.**, San Francisco, CA (US)

(72) Inventors: **George Apostolopoulos**, San Jose, CA (US); **Zhuxuan Jin**, Sunnyvale, CA (US)

(73) Assignee: **Splunk Inc.**, San Francisco, CA (US)

(21) Appl. No.: **16/051,001**

(22) Filed: **Jul. 31, 2018**

Publication Classification

(51) **Int. Cl.**

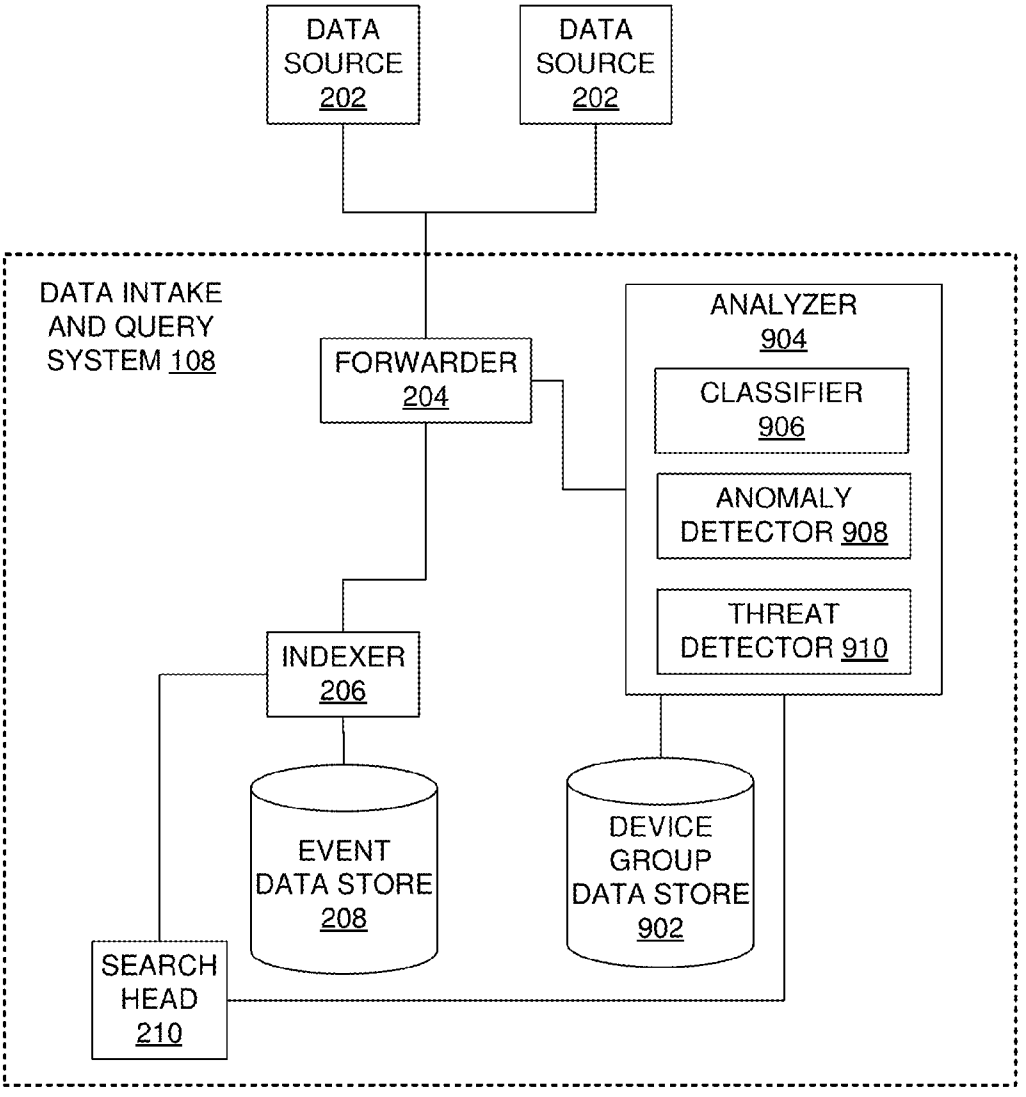
H04L 12/24	(2006.01)
H04L 29/06	(2006.01)
H04L 29/08	(2006.01)
G06K 9/62	(2006.01)
H04L 12/26	(2006.01)

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

CPC ***H04L 41/0893*** (2013.01); ***H04L 41/145*** (2013.01); ***H04L 63/1425*** (2013.01); ***H04L 41/082*** (2013.01); ***H04L 43/02*** (2013.01); ***G06K 9/6278*** (2013.01); ***H04L 43/04*** (2013.01); ***H04L 41/22*** (2013.01); ***H04L 67/303*** (2013.01)

(57) **ABSTRACT**

One or more embodiments are directed behavioral based device clustering. A network traffic log of devices in the network is received. Features of devices are extracted from the network traffic log and aggregated into an aggregated feature matrix on a per device basis. By applying a topic modeling algorithm to the aggregated feature matrix, the devices are clustered into device groups according to behavior groups. A device is assigned to the device group to create an assignment.



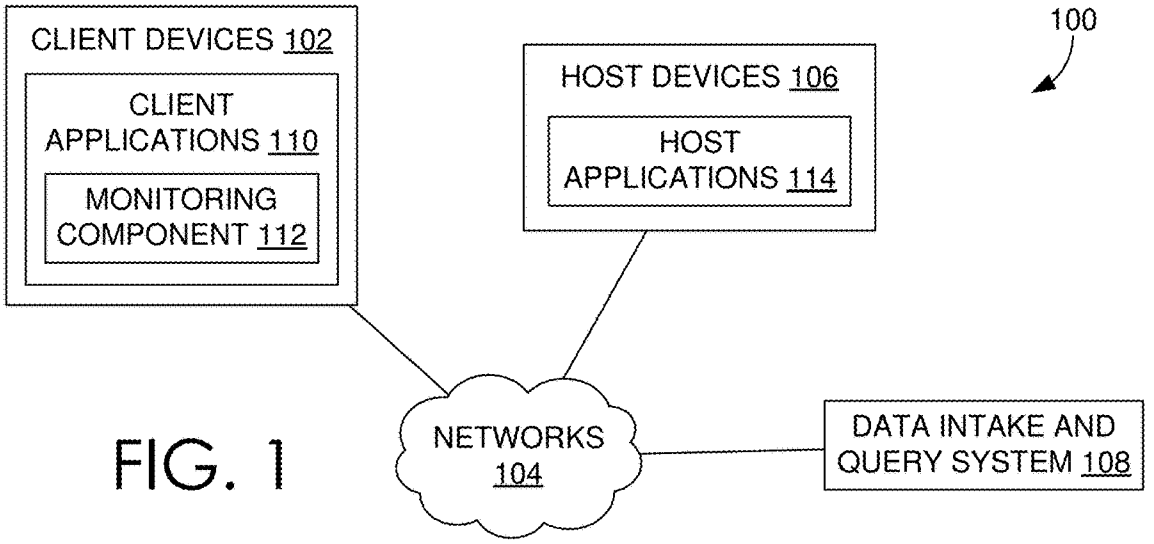

100
CLIENT DEVICES 102
CLIENT APPLICATIONS 110
MONITORING COMPONENT 112
HOST DEVICES 106
HOST APPLICATIONS 114
NETWORKS 104
DATA INTAKE AND QUERY SYSTEM 108

FIG. 1

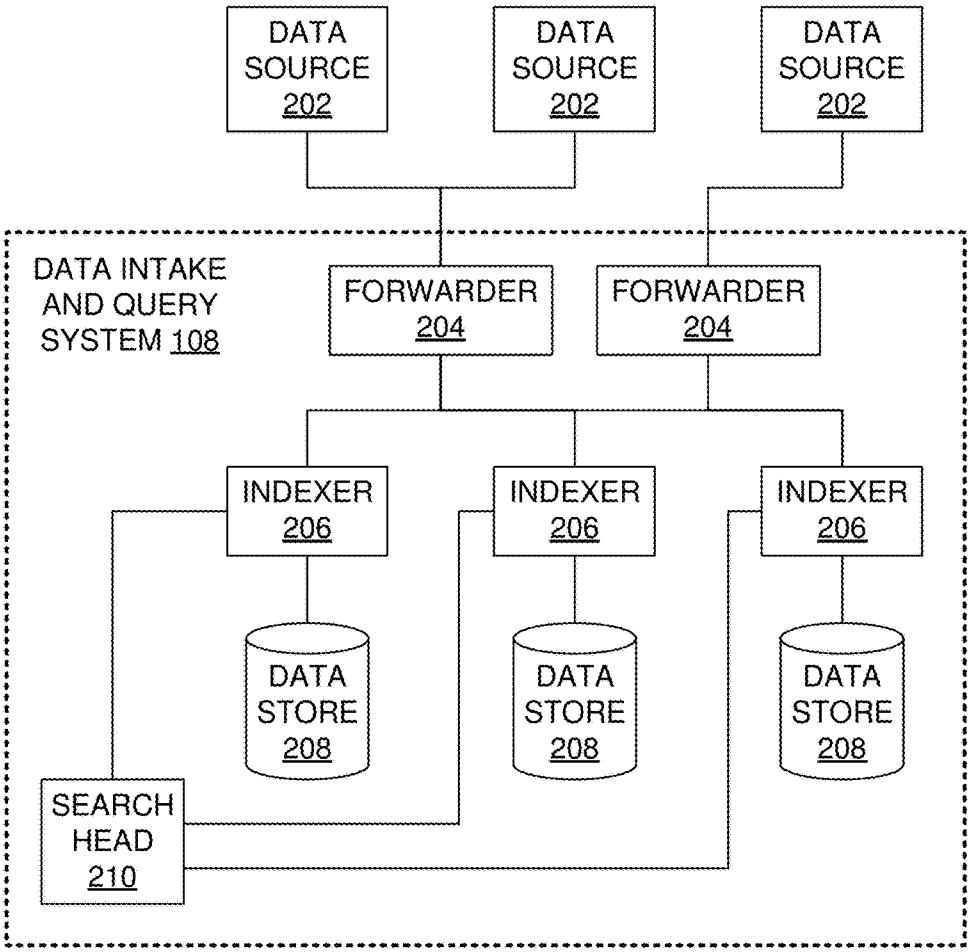

DATA SOURCE 202
DATA SOURCE 202
DATA SOURCE 202
DATA INTAKE AND QUERY SYSTEM 108
FORWARDER 204
FORWARDER 204
INDEXER 206
INDEXER 206
INDEXER 206
DATA STORE 208
DATA STORE 208
DATA STORE 208
SEARCH HEAD 210

FIG. 2

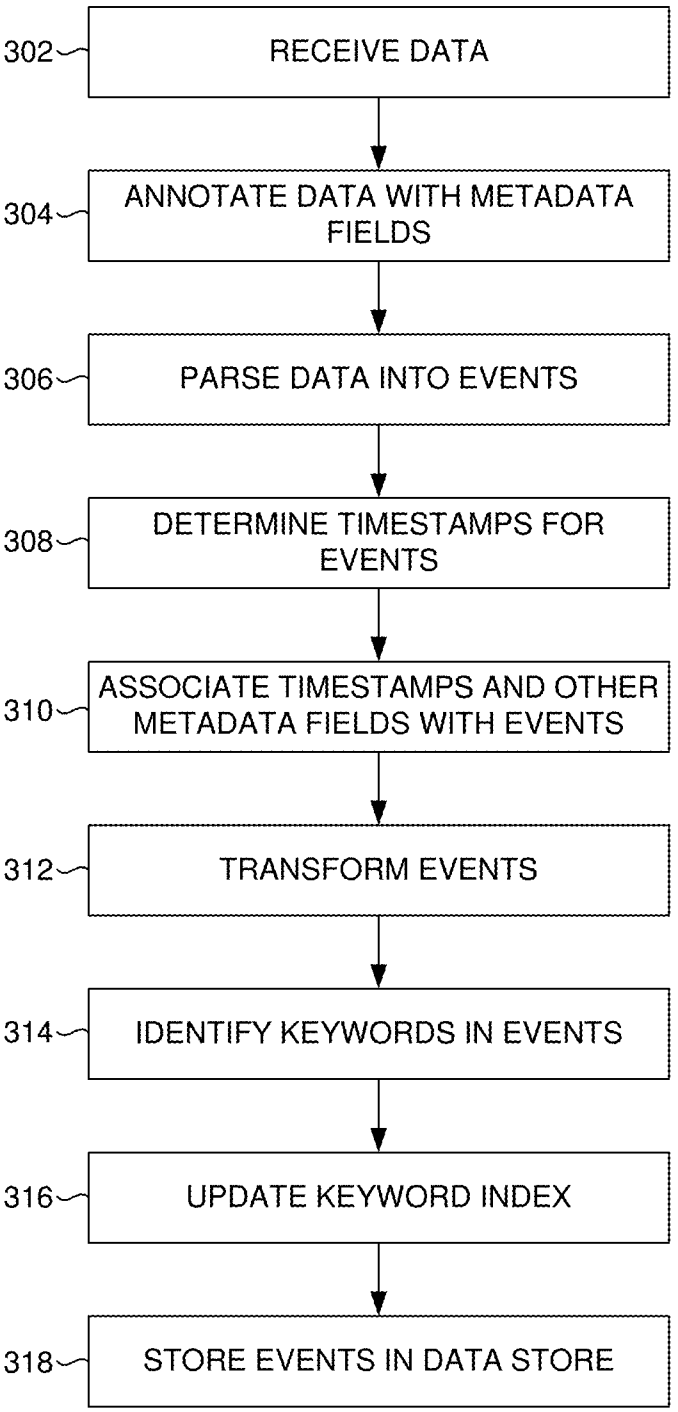

302
RECEIVE DATA
304
ANNOTATE DATA WITH METADATA FIELDS
306
PARSE DATA INTO EVENTS
308
DETERMINE TIMESTAMPS FOR EVENTS
310
ASSOCIATE TIMESTAMPS AND OTHER METADATA FIELDS WITH EVENTS
312
TRANSFORM EVENTS
314
IDENTIFY KEYWORDS IN EVENTS
316
UPDATE KEYWORD INDEX
318
STORE EVENTS IN DATA STORE

FIG. 3

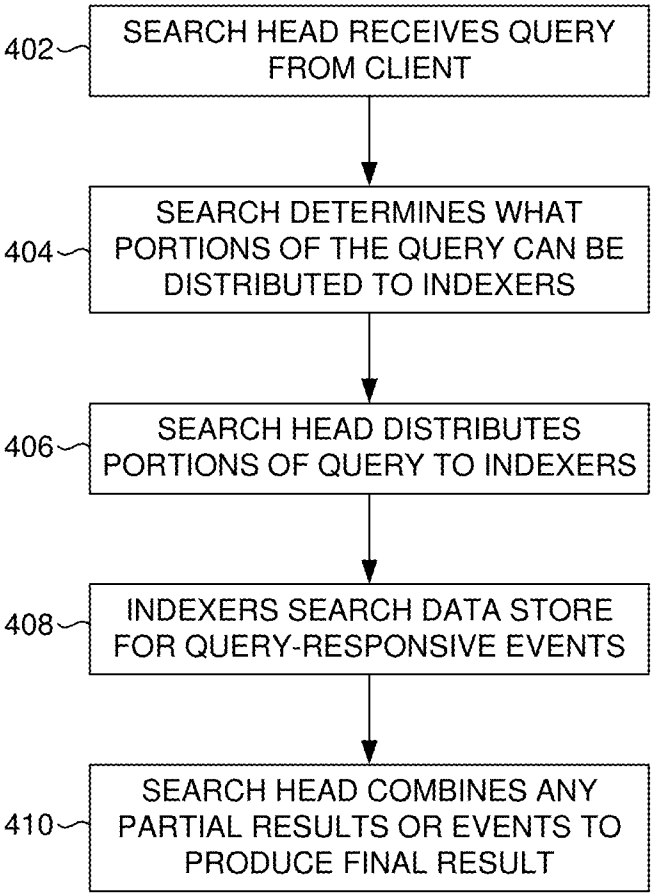

402
SEARCH HEAD RECEIVES QUERY FROM CLIENT
404
SEARCH DETERMINES WHAT PORTIONS OF THE QUERY CAN BE DISTRIBUTED TO INDEXERS
406
SEARCH HEAD DISTRIBUTES PORTIONS OF QUERY TO INDEXERS
408
INDEXERS SEARCH DATA STORE FOR QUERY-RESPONSIVE EVENTS
410
SEARCH HEAD COMBINES ANY PARTIAL RESULTS OR EVENTS TO PRODUCE FINAL RESULT

FIG. 4

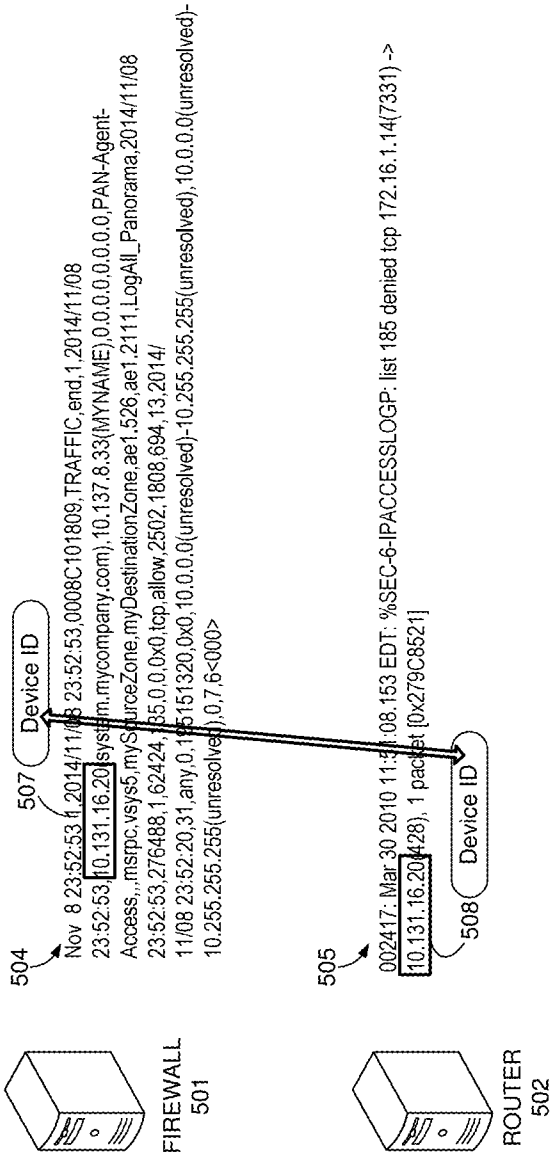

Device ID
507
504
Nov 8 23:52:53 1 2014/11/08 23:52:53,0008C101809,TRAFFIC,end,1,2014/11/08
23:52:53,10.131.16.20,system.mycompany.com),10.137.8.33(MYNAME),0.0.0.0,0.0.0.0,PAN-Agent-
Access,,,msrpc,vsys5,mySourceZone,myDestinationZone,ae1.526,ae1.2111,LogAll_Panorama,2014/11/08
23:52:53,276488,1,62424,135,0,0,0x0,tcp,allow,2502,1808,694,13,2014/
11/08 23:52:20,31,any,0,1[illegible]151320,0x0,10.0.0.0(unresolved),10.0.0.0(unresolved)-
10.255.255.255(unresolved),0,7,6<000>
505
002417: Mar 30 2010 11:[illegible]:08.153 EDT: %SEC-6-IPACCESSLOGP: list 185 denied tcp 172.16.1.14(7331) ->
10.131.16.20(4428), 1 packet [0x279C8521]
508
Device ID
FIREWALL
501
ROUTER
502

FIG. 5

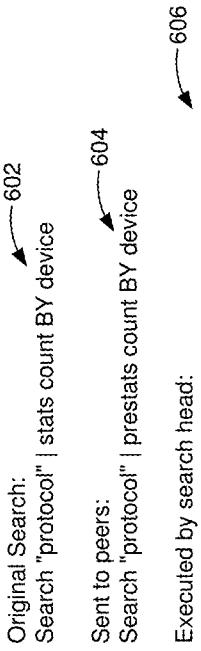

Original Search:
Search "protocol" | stats count BY device
602
Sent to peers:
Search "protocol" | prestats count BY device
604
Executed by search head:
Aggregate the prestats results received from peers
606

FIG. 6

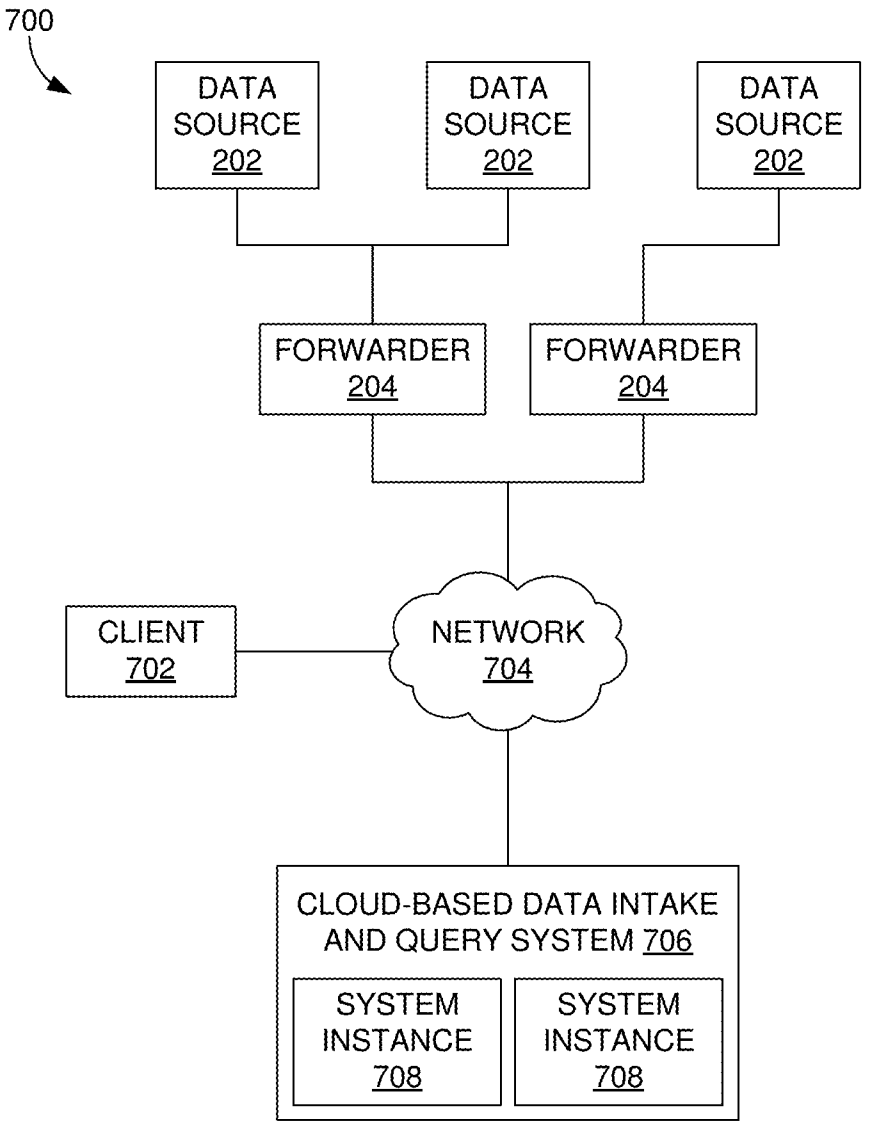

700
DATA SOURCE 202
DATA SOURCE 202
DATA SOURCE 202
FORWARDER 204
FORWARDER 204
CLIENT 702
NETWORK 704
CLOUD-BASED DATA INTAKE AND QUERY SYSTEM 706
SYSTEM INSTANCE 708
SYSTEM INSTANCE 708

FIG. 7

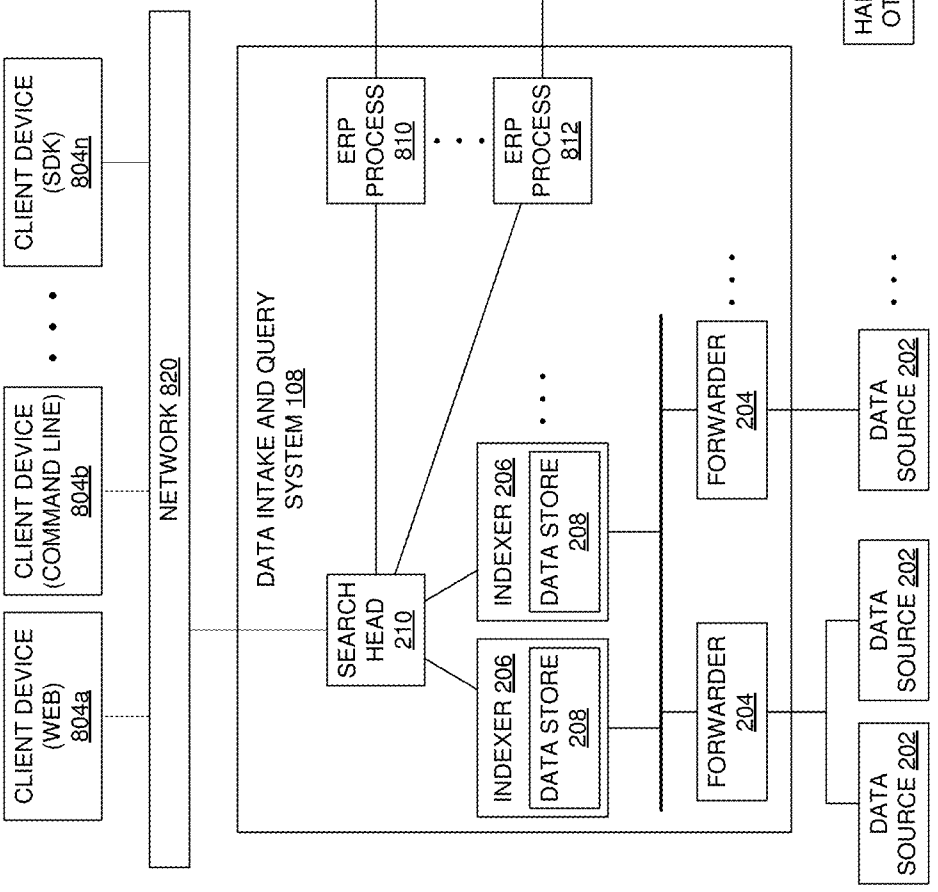

CLIENT DEVICE (WEB) 804a
CLIENT DEVICE (COMMAND LINE) 804b
CLIENT DEVICE (SDK) 804n
NETWORK 820
DATA INTAKE AND QUERY SYSTEM 108
SEARCH HEAD 210
ERP PROCESS 810
ERP PROCESS 812
INDEXER 206
DATA STORE 208
INDEXER 206
DATA STORE 208
FORWARDER 204
FORWARDER 204
HADOOP OR OTHER 814
RDBMS 816
DATA SOURCE 202
DATA SOURCE 202
DATA SOURCE 202

FIG. 8

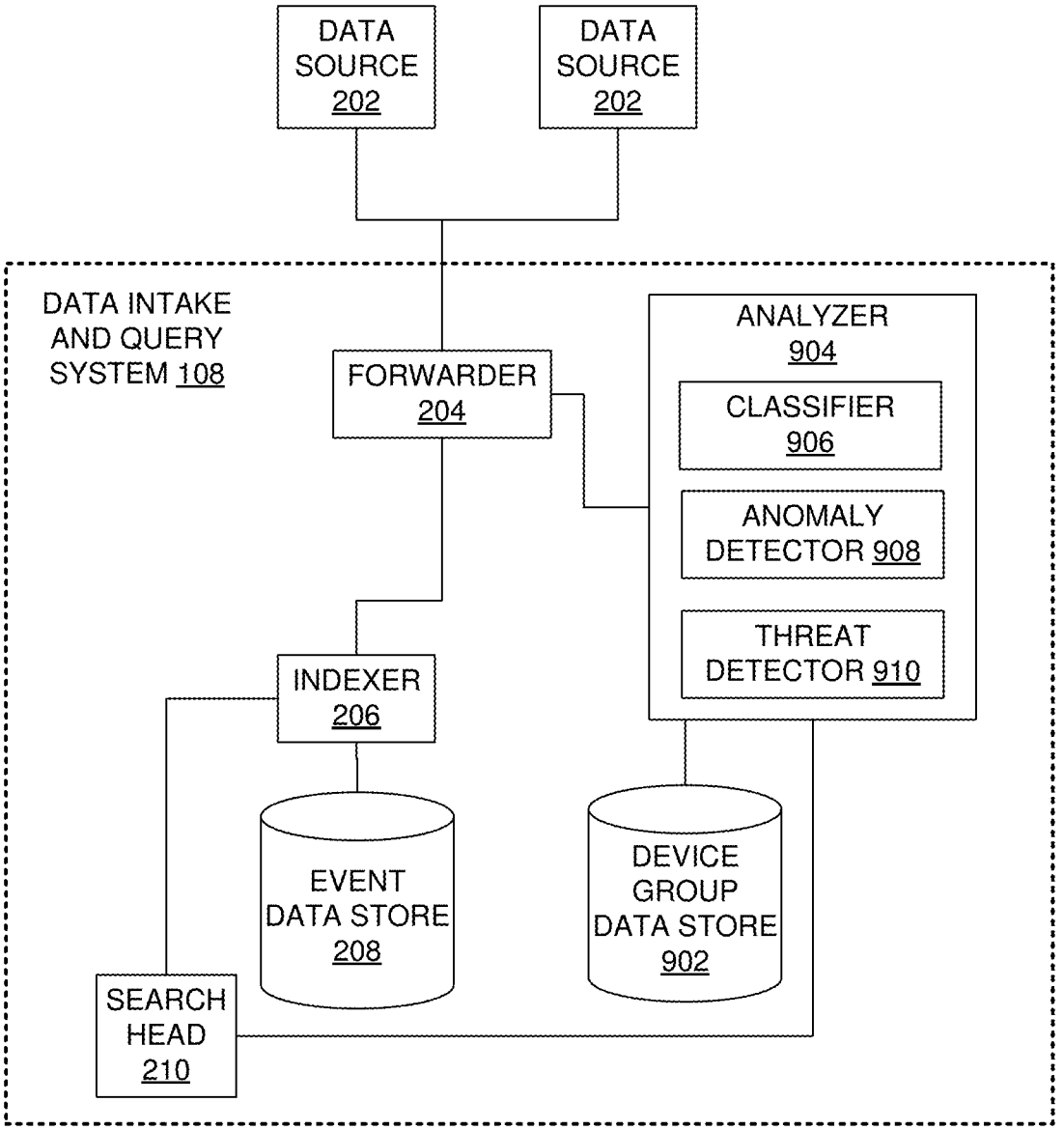

DATA SOURCE 202
DATA SOURCE 202
DATA INTAKE AND QUERY SYSTEM 108
FORWARDER 204
ANALYZER 904
CLASSIFIER 906
ANOMALY DETECTOR 908
THREAT DETECTOR 910
INDEXER 206
EVENT DATA STORE 208
DEVICE GROUP DATA STORE 902
SEARCH HEAD 210

FIG. 9

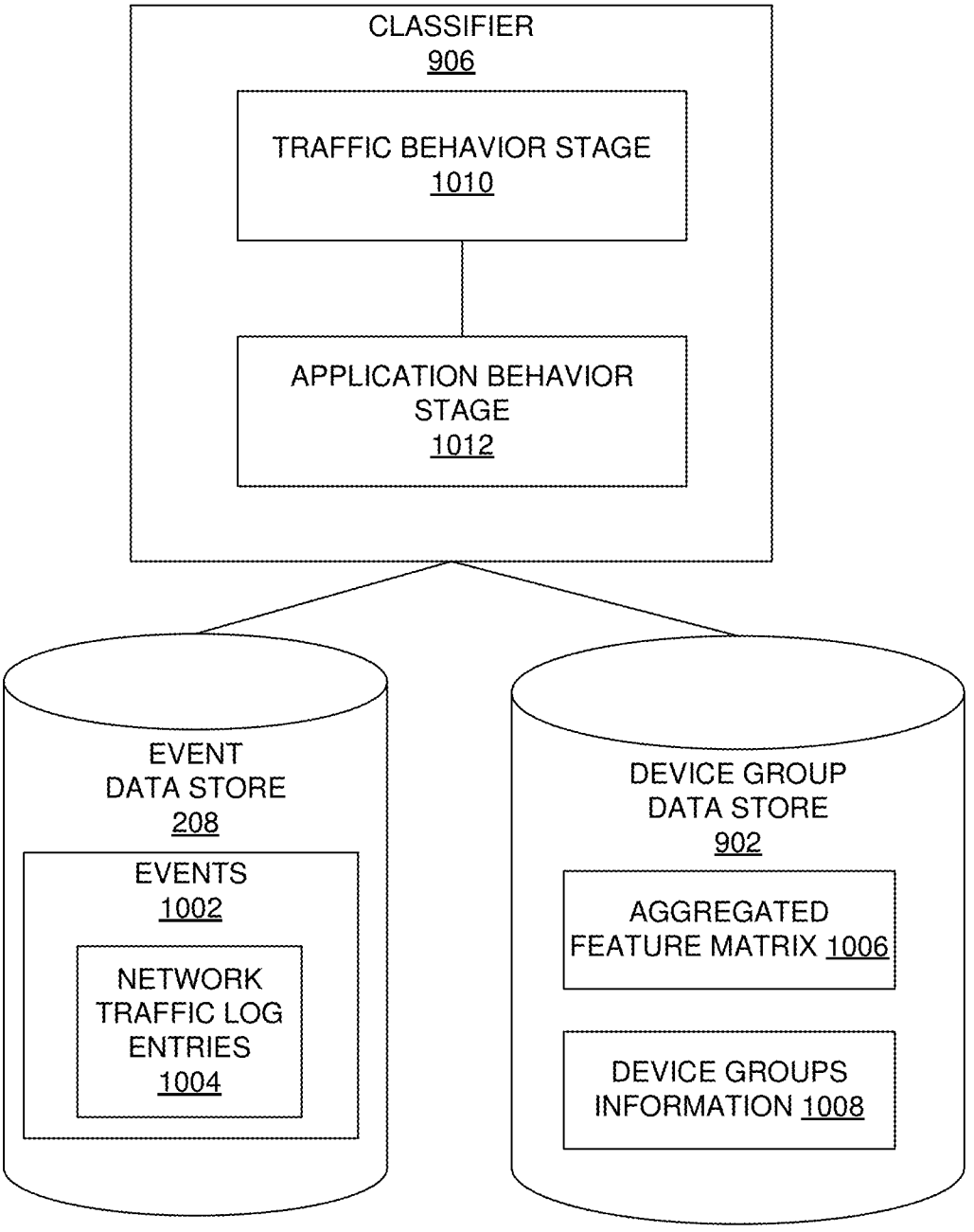

CLASSIFIER
906
TRAFFIC BEHAVIOR STAGE
1010
APPLICATION BEHAVIOR STAGE
1012
EVENT DATA STORE
208
EVENTS
1002
NETWORK TRAFFIC LOG ENTRIES
1004
DEVICE GROUP DATA STORE
902
AGGREGATED FEATURE MATRIX 1006
DEVICE GROUPS INFORMATION 1008

FIG. 10

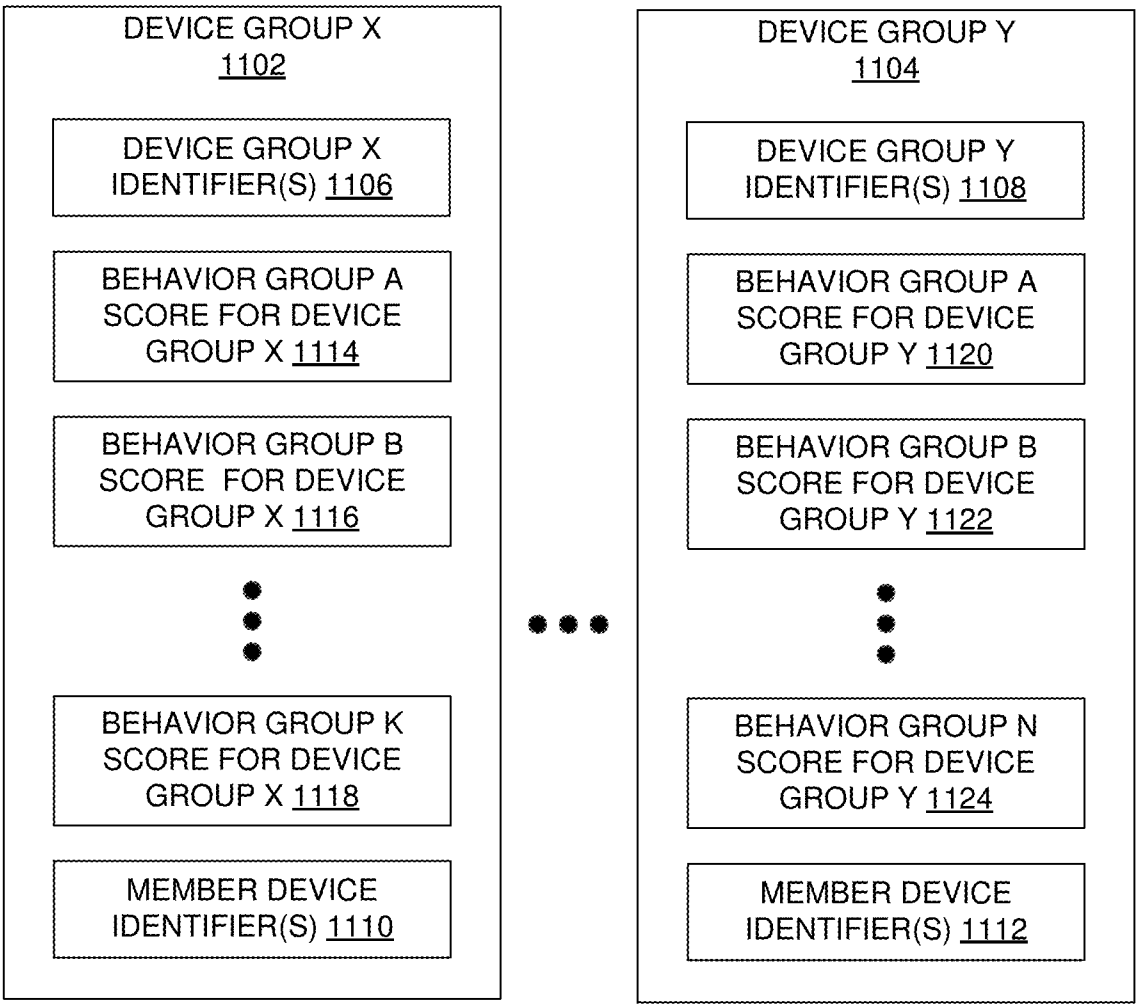

DEVICE GROUP X
1102
DEVICE GROUP X IDENTIFIER(S) 1106
BEHAVIOR GROUP A SCORE FOR DEVICE GROUP X 1114
BEHAVIOR GROUP B SCORE FOR DEVICE GROUP X 1116
BEHAVIOR GROUP K SCORE FOR DEVICE GROUP X 1118
MEMBER DEVICE IDENTIFIER(S) 1110
DEVICE GROUP Y
1104
DEVICE GROUP Y IDENTIFIER(S) 1108
BEHAVIOR GROUP A SCORE FOR DEVICE GROUP Y 1120
BEHAVIOR GROUP B SCORE FOR DEVICE GROUP Y 1122
BEHAVIOR GROUP N SCORE FOR DEVICE GROUP Y 1124
MEMBER DEVICE IDENTIFIER(S) 1112

FIG. 11

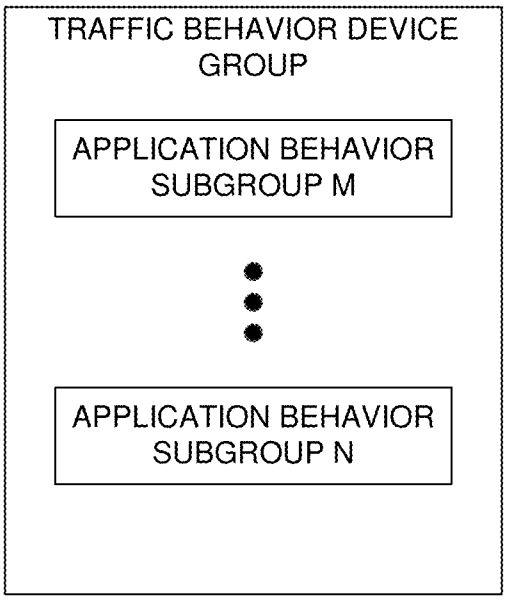

TRAFFIC BEHAVIOR DEVICE GROUP
APPLICATION BEHAVIOR SUBGROUP M
APPLICATION BEHAVIOR SUBGROUP N

FIG. 12

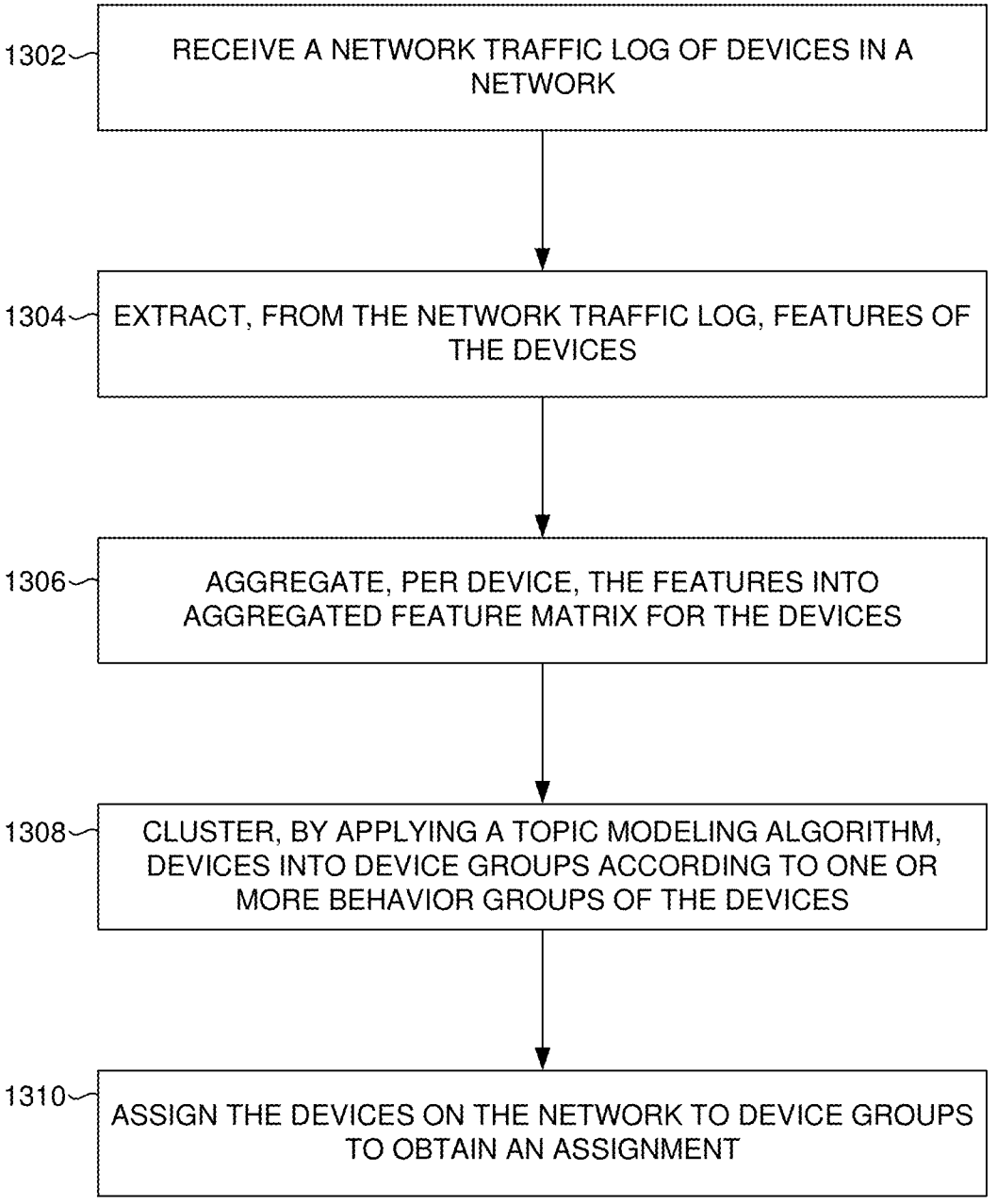

1302
RECEIVE A NETWORK TRAFFIC LOG OF DEVICES IN A NETWORK
1304
EXTRACT, FROM THE NETWORK TRAFFIC LOG, FEATURES OF THE DEVICES
1306
AGGREGATE, PER DEVICE, THE FEATURES INTO AGGREGATED FEATURE MATRIX FOR THE DEVICES
1308
CLUSTER, BY APPLYING A TOPIC MODELING ALGORITHM, DEVICES INTO DEVICE GROUPS ACCORDING TO ONE OR MORE BEHAVIOR GROUPS OF THE DEVICES
1310
ASSIGN THE DEVICES ON THE NETWORK TO DEVICE GROUPS TO OBTAIN AN ASSIGNMENT

FIG. 13

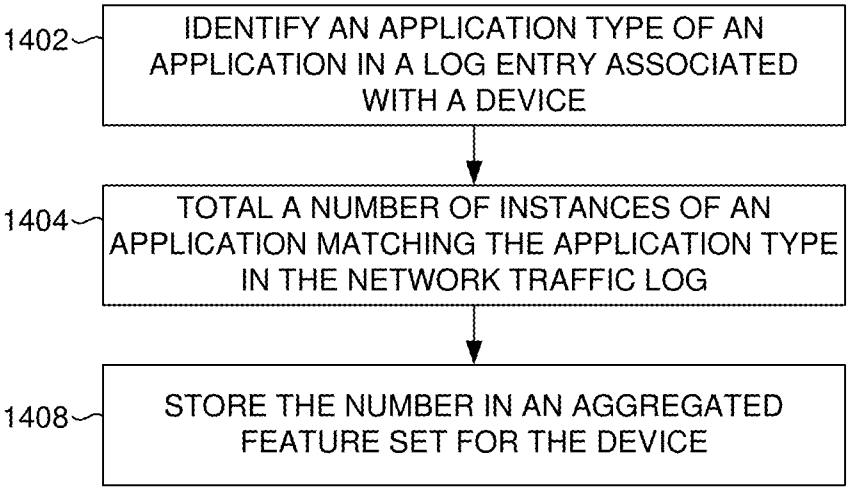

1402
IDENTIFY AN APPLICATION TYPE OF AN APPLICATION IN A LOG ENTRY ASSOCIATED WITH A DEVICE
1404
TOTAL A NUMBER OF INSTANCES OF AN APPLICATION MATCHING THE APPLICATION TYPE IN THE NETWORK TRAFFIC LOG
1408
STORE THE NUMBER IN AN AGGREGATED FEATURE SET FOR THE DEVICE

FIG. 14

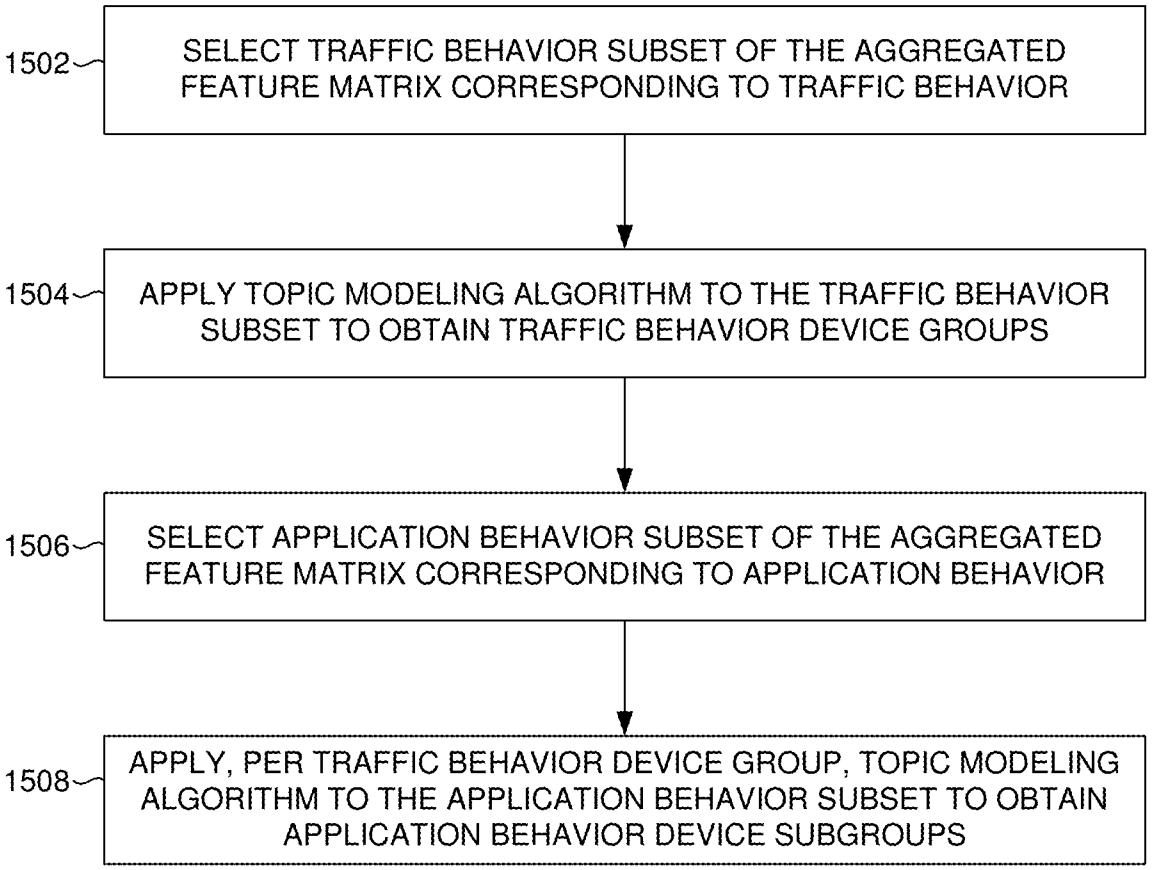

1502
SELECT TRAFFIC BEHAVIOR SUBSET OF THE AGGREGATED FEATURE MATRIX CORRESPONDING TO TRAFFIC BEHAVIOR
1504
APPLY TOPIC MODELING ALGORITHM TO THE TRAFFIC BEHAVIOR SUBSET TO OBTAIN TRAFFIC BEHAVIOR DEVICE GROUPS
1506
SELECT APPLICATION BEHAVIOR SUBSET OF THE AGGREGATED FEATURE MATRIX CORRESPONDING TO APPLICATION BEHAVIOR
1508
APPLY, PER TRAFFIC BEHAVIOR DEVICE GROUP, TOPIC MODELING ALGORITHM TO THE APPLICATION BEHAVIOR SUBSET TO OBTAIN APPLICATION BEHAVIOR DEVICE SUBGROUPS

FIG. 15

DEVICE GROUP 3 1602

DEVICE GROUP 3 BEHAVIOR GROUPS SCORES:
0.35 - trafficVolumeSrcIn2In
0.56 - appOfdomainControllerDst
0.42 – appOfauthserviceDst
1604

DEVICE GROUP 3 MEMBER IDENTIFIERS:
10.121.10.11 (SCL-VS-DMZDC02)
10.118.0.11 (SCL-VS-DMZDC01)
10.134.0.11 (SAL-VS-DMZDC01)
1606

DEVICE GROUP 8 1610

DEVICE GROUP 8 BEHAVIOR GROUPS SCORES:
0.35 - trafficVolumeSrcIn2In
0.98 – appOfdatabaseDst
1612

DEVICE GROUP 8 MEMBER IDENTIFIERS:
10.115.30.31 (scl-s-spotdb01.corp.svbank.com)
10.115.30.38 (scl-s-sqlc1n2.corp.svbank.com)
10.122.22.24 (obodbl7ads.corp.svbank.com),
10.122.6.43 (obodbl8ads.corp.svbank.com)
1614

FIG. 16

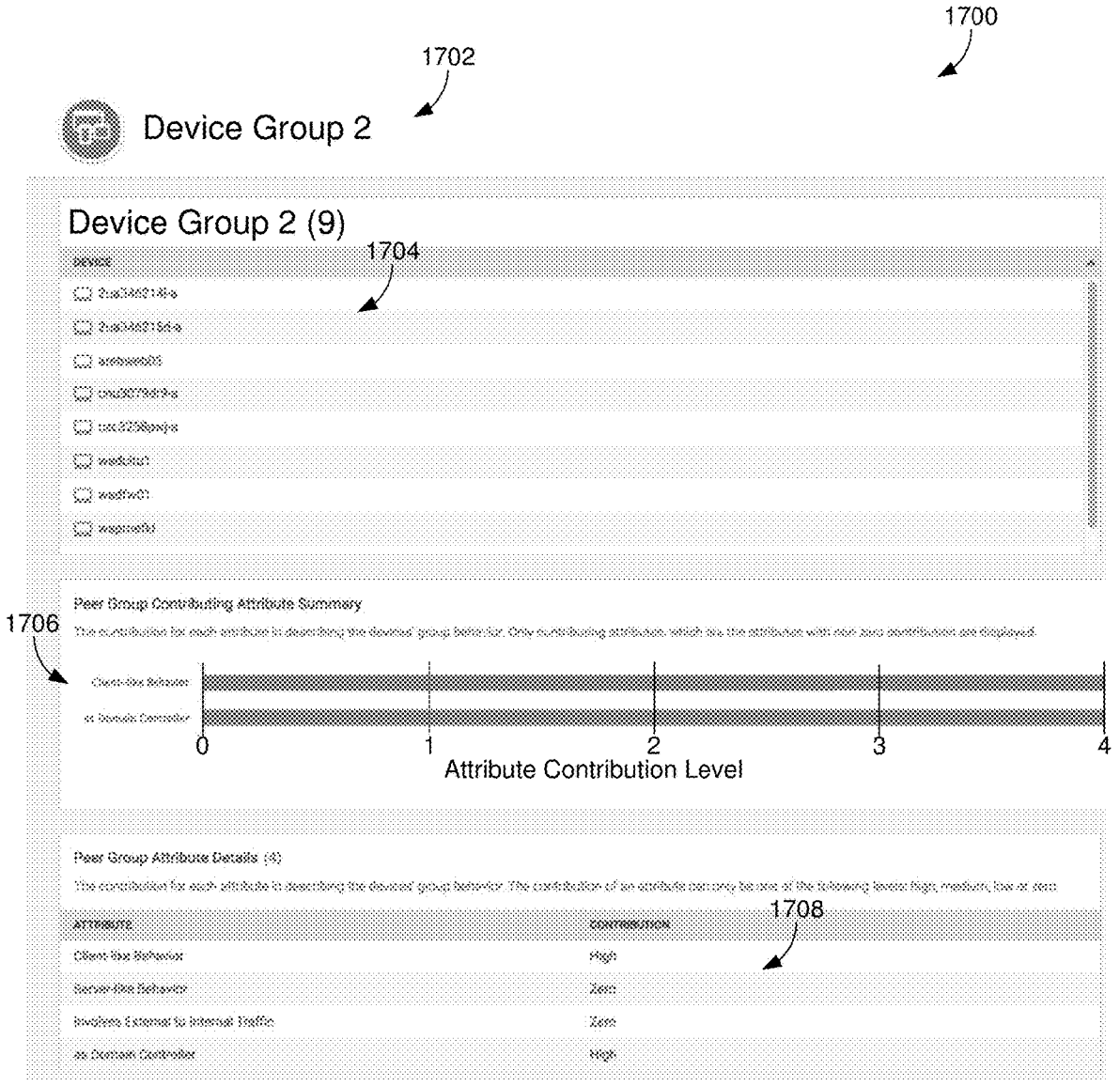

1700
1702
Device Group 2
Device Group 2 (9)
1704
1706
Peer Group Contributing Attribute Summary
0
1
2
3
4
Attribute Contribution Level
Peer Group Attribute Details (4)
1708

FIG. 17

BEHAVIORAL BASED DEVICE CLUSTERING SYSTEM AND METHOD

BACKGROUND

[0001] Network systems are a set of interconnected devices that together provide computing functionality. For example, networks may include storage devices, end-user terminals, application servers, routers, and other devices. In order for the continual operation of the network, the network is managed by managing the devices of the network. Management of a network includes performing hardware and software updates, allocating resources, and performing intrusion detection. Because of the number of devices in a network, the devices are often grouped and managed together. Generally, to group devices, a network administrator manually groups devices based on a static attribute (e.g., role of person using device or type of device) of the device.

BRIEF DESCRIPTION OF DRAWINGS

[0002] In the drawings:

[0003] FIG. **1** illustrates a networked computer environment in which an embodiment may be implemented;

[0004] FIG. **2** illustrates a block diagram of an example data intake and query system in which an embodiment may be implemented;

[0005] FIG. **3** is a flow diagram that illustrates how indexers process, index, and store data received from forwarders in accordance with the disclosed embodiments;

[0006] FIG. **4** is a flow diagram that illustrates how a search head and indexers perform a search query in accordance with the disclosed embodiments;

[0007] FIG. **5** illustrates a scenario where a common customer ID is found among log data received from three disparate sources in accordance with the disclosed embodiments;

[0008] FIG. **6** illustrates an example search query received from a client and executed by search peers in accordance with the disclosed embodiments;

[0009] FIG. **7** illustrates a block diagram of an example cloud-based data intake and query system in which an embodiment may be implemented;

[0010] FIG. **8** illustrates a block diagram of an example data intake and query system that performs searches across external data systems in accordance with the disclosed embodiments;

[0011] FIG. **9** illustrates a block diagram of an example data intake and query system in which an embodiment may be implemented;

[0012] FIG. **10** illustrates a block diagram of an example classifier and data store in which an embodiment may be implemented;

[0013] FIG. **11** illustrates a block diagram of device groups in accordance with disclosed embodiments;

[0014] FIG. **12** illustrates a block diagram of nesting device groups in accordance with disclosed embodiments;

[0015] FIG. **13** is a flow diagram that illustrates how devices may be groups according to behavior in accordance with the disclosed embodiments;

[0016] FIG. **14** is a flow diagram that illustrates how features may be extracted based on application type in accordance with the disclosed embodiments;

[0017] FIG. **15** is a flow diagram that illustrates how to create nested device groups in accordance with disclosed embodiments;

[0018] FIG. **16** is an example diagram of a device group clustering in accordance with disclosed embodiments; and

[0019] FIG. **17** illustrates an example Graphical User Interface in accordance with disclosed embodiments.

DETAILED DESCRIPTION

[0020] Specific embodiments of the invention will now be described in detail with reference to the accompanying figures Like elements in the various figures are denoted by like reference numerals for consistency.

[0021] In the following detailed description of embodiments of the invention, numerous specific details are set forth in order to provide a more thorough understanding of the invention. However, it will be apparent to one of ordinary skill in the art that the invention may be practiced without these specific details. In other instances, well-known features have not been described in detail to avoid unnecessarily complicating the description.

[0022] Throughout the application, ordinal numbers (e.g., first, second, third, etc.) may be used as an adjective for an element (i.e., any noun in the application). The use of ordinal numbers is not to imply or create any particular ordering of the elements nor to limit any element to being only a single element unless expressly disclosed, such as by the use of the terms "before", "after", "single", and other such terminology. Rather, the use of ordinal numbers is to distinguish between the elements. By way of an example, a first element is distinct from a second element, and the first element may encompass more than one element and succeed (or precede) the second element in an ordering of elements.

[0023] Further, although the description includes a discussion of various embodiments, the various disclosed embodiments may be combined in virtually any manner. All combinations are contemplated herein.

[0024] In general, embodiments are directed to an efficient technique for grouping devices for management based on network behavior of the devices. In particular, one or more embodiments use a network traffic log of the devices in the network. The network traffic log is a recording of traffic events in the network, such as the communication of packets in the network. One or more embodiments extract features, from the network traffic log, of the devices that describe the behavior of the devices. For example, the network traffic log includes features of the devices, such as whether the device is a destination or source of a communication, an application being used, port numbers, packet size, protocols used by the device, etc. By aggregating the features for a device, and using the aggregated features, the behavior of the devices may be used. According to the behavior, the devices are grouped into device groups according to the behavior of the devices.

[0025] Devices in the same device group may be managed together. Specifically, by grouping devices according to behavior of the devices, upgrades, configurations and other management of the devices may be performed more efficiently to increase the efficiency of the network. Additionally, anomaly detection and other security related detection, prevention, and correction may be performed based on the device groups.

[0026] Embodiments are described herein according to the following outline:

[0049] 1.0. General Overview

[0050] Modern data centers and other computing environments can comprise anywhere from a few host computer systems to thousands of systems configured to process data, service requests from remote clients, and perform numerous other computational tasks. During operation, various components within these computing environments often generate significant volumes of machine-generated data. For example, machine data is generated by various components in the information technology (IT) environments, such as servers, sensors, routers, mobile devices, Internet of Things (IoT) devices, etc. Machine-generated data can include system logs, network packet data, sensor data, application program data, error logs, stack traces, system performance data, etc. In general, machine-generated data can also include performance data, diagnostic information, and many other types of data that can be analyzed to diagnose performance problems, monitor user interactions, and to derive other insights.

[0051] A number of tools are available to analyze machine data, that is, machine-generated data. In order to reduce the size of the potentially vast amount of machine data that may be generated, many of these tools typically pre-process the data based on anticipated data-analysis needs. For example, pre-specified data items may be extracted from the machine data and stored in a database to facilitate efficient retrieval and analysis of those data items at search time. However, the rest of the machine data typically is not saved and discarded during pre-processing. As storage capacity becomes progressively cheaper and more plentiful, there are fewer incentives to discard these portions of machine data and many reasons to retain more of the data.

[0052] This plentiful storage capacity is presently making it feasible to store massive quantities of minimally processed machine data for later retrieval and analysis. In general, storing minimally processed machine data and performing analysis operations at search time can provide greater flexibility because it enables an analyst to search all of the machine data, instead of searching only a pre-specified set of data items. This may enable an analyst to investigate different aspects of the machine data that previously were unavailable for analysis.

[0053] However, analyzing and searching massive quantities of machine data presents a number of challenges. For example, a data center, servers, or network appliances may generate many different types and formats of machine data (e.g., system logs, network packet data (e.g., wire data, etc.), sensor data, application program data, error logs, stack traces, system performance data, operating system data, virtualization data, etc.) from thousands of different components, which can collectively be very time-consuming to analyze. In another example, mobile devices may generate large amounts of information relating to data accesses, application performance, operating system performance, network performance, etc. There can be millions of mobile devices that report these types of information.

[0054] These challenges can be addressed by using an event-based data intake and query system, such as the SPLUNK® ENTERPRISE system developed by Splunk Inc. of San Francisco, Calif. The SPLUNK® ENTERPRISE system is the leading platform for providing real-time operational intelligence that enables organizations to collect, index, and search machine-generated data from various websites, applications, servers, networks, and mobile devices that power their businesses. The SPLUNK® ENTERPRISE system is particularly useful for analyzing data which is commonly found in system log files, network data, and other data input sources. Although many of the techniques described herein are explained with reference to a data intake and query system similar to the SPLUNK® ENTERPRISE system, these techniques are also applicable to other types of data systems.

[0055] In the SPLUNK® ENTERPRISE system, machine-generated data are collected and stored as "events". An event comprises a portion of the machine-generated data and is associated with a specific point in time. For example, events may be derived from "time series data," where the time series data comprises a sequence of data points (e.g., performance measurements from a computer system, etc.) that are associated with successive points in time. In general, each event can be associated with a timestamp that is derived from the raw data in the event, determined through interpolation between temporally proximate events having known timestamps, or determined based on other configurable rules for associating timestamps with events, etc.

[0056] In some instances, machine data can have a predefined format, where data items with specific data formats are stored at predefined locations in the data. For example, the machine data may include data stored as fields in a database table. In other instances, machine data may not have a predefined format, that is, the data is not at fixed, predefined locations, but the data does have repeatable patterns and is not random. This means that some machine data can comprise various data items of different data types and that may be stored at different locations within the data. For example, when the data source is an operating system

log, an event can include one or more lines from the operating system log containing raw data that includes different types of performance and diagnostic information associated with a specific point in time.

[0057] Examples of components which may generate machine data from which events can be derived include, but are not limited to, web servers, application servers, databases, firewalls, routers, operating systems, and software applications that execute on computer systems, mobile devices, sensors, Internet of Things (IoT) devices, etc. The data generated by such data sources can include, for example and without limitation, server log files, activity log files, configuration files, messages, network packet data, performance measurements, sensor measurements, etc.

[0058] The SPLUNK® ENTERPRISE system uses flexible schema to specify how to extract information from the event data. A flexible schema may be developed and redefined as needed. Note that a flexible schema may be applied to event data "on the fly," when it is needed (e.g., at search time, index time, ingestion time, etc.). When the schema is not applied to event data until search time it may be referred to as a "late-binding schema."

[0059] During operation, the SPLUNK® ENTERPRISE system starts with raw input data (e.g., one or more system logs, streams of network packet data, sensor data, application program data, error logs, stack traces, system performance data, etc.). The system divides this raw data into blocks (e.g., buckets of data, each associated with a specific time frame, etc.), and parses the raw data to produce timestamped events. The system stores the timestamped events in a data store. The system enables users to run queries against the stored data to, for example, retrieve events that meet criteria specified in a query, such as containing certain keywords or having specific values in defined fields. As used herein throughout, data that is part of an event is referred to as "event data". In this context, the term "field" refers to a location in the event data containing one or more values for a specific data item. As will be described in more detail herein, the fields are defined by extraction rules (e.g., regular expressions) that derive one or more values from the portion of raw machine data in each event that has a particular field specified by an extraction rule. The set of values so produced are semantically-related (such as IP address), even though the raw machine data in each event may be in different formats (e.g., semantically-related values may be in different positions in the events derived from different sources).

[0060] As noted above, the SPLUNK® ENTERPRISE system utilizes a late-binding schema to event data while performing queries on events. One aspect of a late-binding schema is applying "extraction rules" to event data to extract values for specific fields during search time. More specifically, the extraction rules for a field can include one or more instructions that specify how to extract a value for the field from the event data. An extraction rule can generally include any type of instruction for extracting values from data in events. In some cases, an extraction rule comprises a regular expression where a sequence of characters form a search pattern, in which case the rule is referred to as a "regex rule." The system applies the regex rule to the event data to extract values for associated fields in the event data by searching the event data for the sequence of characters defined in the regex rule.

[0061] In the SPLUNK® ENTERPRISE system, a field extractor may be configured to automatically generate extraction rules for certain field values in the events when the events are being created, indexed, or stored, or possibly at a later time. Alternatively, a user may manually define extraction rules for fields using a variety of techniques. In contrast to a conventional schema for a database system, a late-binding schema is not defined at data ingestion time. Instead, the late-binding schema can be developed on an ongoing basis until the time a query is actually executed. This means that extraction rules for the fields in a query may be provided in the query itself or may be located during execution of the query. Hence, as a user learns more about the data in the events, the user can continue to refine the late-binding schema by adding new fields, deleting fields, or modifying the field extraction rules for use the next time the schema is used by the system. Because the SPLUNK® ENTERPRISE system maintains the underlying raw data and uses late-binding schema for searching the raw data, it enables a user to continue investigating and learn valuable insights about the raw data.

[0062] In some embodiments, a common field name may be used to reference two or more fields containing equivalent data items, even though the fields may be associated with different types of events that possibly have different data formats and different extraction rules. By enabling a common field name to be used to identify equivalent fields from different types of events generated by disparate data sources, the system facilitates use of a "common information model" (CIM) across the disparate data sources (further discussed with respect to FIG. **5**).

[0063] 2.0. Operating Environment

[0064] FIG. **1** illustrates a networked computer system **100** in which an embodiment may be implemented. Those skilled in the art would understand that FIG. **1** represents one example of a networked computer system and other embodiments may use different arrangements.

[0065] The networked computer system **100** comprises one or more computing devices. These one or more computing devices comprise any combination of hardware and software configured to implement the various logical components described herein. For example, the one or more computing devices may include one or more memories that store instructions for implementing the various components described herein, one or more hardware processors configured to execute the instructions stored in the one or more memories, and various data repositories in the one or more memories for storing data structures utilized and manipulated by the various components.

[0066] In an embodiment, one or more client devices **102** are coupled to one or more host devices **106** and a data intake and query system **108** via one or more networks **104**. Networks **104** broadly represent one or more LANs, WANs, cellular networks (e.g., LTE, HSPA, 3G, and other cellular technologies), and/or networks using any of wired, wireless, terrestrial microwave, or satellite links, and may include the public Internet.

[0067] 2.1. Host Devices

[0068] In the illustrated embodiment, a system **100** includes one or more host devices **106**. Host devices **106** may broadly include any number of computers, virtual machine instances, and/or data centers that are configured to host or execute one or more instances of host applications **114**. In general, a host device **106** may be involved, directly

or indirectly, in processing requests received from client devices **102**. Each host device **106** may comprise, for example, one or more of a network device, a web server, an application server, a database server, etc. A collection of host devices **106** may be configured to implement a network-based service. For example, a provider of a network-based service may configure one or more host devices **106** and host applications **114** (e.g., one or more web servers, application servers, database servers, etc.) to collectively implement the network-based application.

[0069] In general, client devices **102** communicate with one or more host applications **114** to exchange information. The communication between a client device **102** and a host application **114** may, for example, be based on the Hypertext Transfer Protocol (HTTP) or any other network protocol. Content delivered from the host application **114** to a client device **102** may include, for example, HTML documents, media content, etc. The communication between a client device **102** and host application **114** may include sending various requests and receiving data packets. For example, in general, a client device **102** or application running on a client device may initiate communication with a host application **114** by making a request for a specific resource (e.g., based on an HTTP request), and the application server may respond with the requested content stored in one or more response packets.

[0070] In the illustrated embodiment, one or more of host applications **114** may generate various types of performance data during operation, including event logs, network data, sensor data, and other types of machine-generated data. For example, a host application **114** comprising a web server may generate one or more web server logs in which details of interactions between the web server and any number of client devices **102** is recorded. As another example, a host device **106** comprising a router may generate one or more router logs that record information related to network traffic managed by the router. As yet another example, a host application **114** comprising a database server may generate one or more logs that record information related to requests sent from other host applications **114** (e.g., web servers or application servers) for data managed by the database server.

[0071] 2.2. Client Devices

[0072] Client devices **102** of FIG. **1** represent any computing device capable of interacting with one or more host devices **106** via a network **104**. Examples of client devices **102** may include, without limitation, smart phones, tablet computers, handheld computers, wearable devices, laptop computers, desktop computers, servers, portable media players, gaming devices, and so forth. In general, a client device **102** can provide access to different content, for instance, content provided by one or more host devices **106**, etc. Each client device **102** may comprise one or more client applications **110**, described in more detail in a separate section hereinafter.

[0073] 2.3. Client Device Applications

[0074] In an embodiment, each client device **102** may host or execute one or more client applications **110** that are capable of interacting with one or more host devices **106** via one or more networks **104**. For instance, a client application **110** may be or comprise a web browser that a user may use to navigate to one or more websites or other resources provided by one or more host devices **106**. As another example, a client application **110** may comprise a mobile application or "app." For example, an operator of a network-based service hosted by one or more host devices **106** may make available one or more mobile apps that enable users of client devices **102** to access various resources of the network-based service. As yet another example, client applications **110** may include background processes that perform various operations without direct interaction from a user. A client application **110** may include a "plug-in" or "extension" to another application, such as a web browser plug-in or extension.

[0075] In an embodiment, a client application **110** may include a monitoring component **112**. At a high level, the monitoring component **112** comprises a software component or other logic that facilitates generating performance data related to a client device's operating state, including monitoring network traffic sent and received from the client device and collecting other device and/or application-specific information. Monitoring component **112** may be an integrated component of a client application **110**, a plug-in, an extension, or any other type of add-on component. Monitoring component **112** may also be a stand-alone process.

[0076] In one embodiment, a monitoring component **112** may be created when a client application **110** is developed, for example, by an application developer using a software development kit (SDK). The SDK may include custom monitoring code that can be incorporated into the code implementing a client application **110**. When the code is converted to an executable application, the custom code implementing the monitoring functionality can become part of the application itself.

[0077] In some cases, an SDK or other code for implementing the monitoring functionality may be offered by a provider of a data intake and query system, such as a system **108**. In such cases, the provider of the system **108** can implement the custom code so that performance data generated by the monitoring functionality is sent to the system **108** to facilitate analysis of the performance data by a developer of the client application or other users.

[0078] In an embodiment, the custom monitoring code may be incorporated into the code of a client application **110** in a number of different ways, such as the insertion of one or more lines in the client application code that call or otherwise invoke the monitoring component **112**. As such, a developer of a client application **110** can add one or more lines of code into the client application **110** to trigger the monitoring component **112** at desired points during execution of the application. Code that triggers the monitoring component may be referred to as a monitor trigger. For instance, a monitor trigger may be included at or near the beginning of the executable code of the client application **110** such that the monitoring component **112** is initiated or triggered as the application is launched, or included at other points in the code that correspond to various actions of the client application, such as sending a network request or displaying a particular interface.

[0079] In an embodiment, the monitoring component **112** may monitor one or more aspects of network traffic sent and/or received by a client application **110**. For example, the monitoring component **112** may be configured to monitor data packets transmitted to and/or from one or more host applications **114**. Incoming and/or outgoing data packets can be read or examined to identify network data contained within the packets, for example, and other aspects of data packets can be analyzed to determine a number of network

performance statistics. Monitoring network traffic may enable information to be gathered particular to the network performance associated with a client application **110** or set of applications.

[0080] In an embodiment, network performance data refers to any type of data that indicates information about the network and/or network performance. Network performance data may include, for instance, a URL requested, a connection type (e.g., HTTP, HTTPS, etc.), a connection start time, a connection end time, an HTTP status code, request length, response length, request headers, response headers, connection status (e.g., completion, response time(s), failure, etc.), and the like. Upon obtaining network performance data indicating performance of the network, the network performance data can be transmitted to a data intake and query system **108** for analysis.

[0081] Upon developing a client application **110** that incorporates a monitoring component **112**, the client application **110** can be distributed to client devices **102**. Applications generally can be distributed to client devices **102** in any manner, or they can be pre-loaded. In some cases, the application may be distributed to a client device **102** via an application marketplace or other application distribution system. For instance, an application marketplace or other application distribution system might distribute the application to a client device based on a request from the client device to download the application.

[0082] Examples of functionality that enables monitoring performance of a client device are described in U.S. patent application Ser. No. 14/524,748, entitled "UTILIZING PACKET HEADERS TO MONITOR NETWORK TRAFFIC IN ASSOCIATION WITH A CLIENT DEVICE", filed on 27 Oct. 2014, and which is hereby incorporated by reference in its entirety for all purposes.

[0083] In an embodiment, the monitoring component **112** may also monitor and collect performance data related to one or more aspects of the operational state of a client application **110** and/or client device **102**. For example, a monitoring component **112** may be configured to collect device performance information by monitoring one or more client device operations, or by making calls to an operating system and/or one or more other applications executing on a client device **102** for performance information. Device performance information may include, for instance, a current wireless signal strength of the device, a current connection type and network carrier, current memory performance information, a geographic location of the device, a device orientation, and any other information related to the operational state of the client device.

[0084] In an embodiment, the monitoring component **112** may also monitor and collect other device profile information including, for example, a type of client device, a manufacturer and model of the device, versions of various software applications installed on the device, and so forth.

[0085] In general, a monitoring component **112** may be configured to generate performance data in response to a monitor trigger in the code of a client application **110** or other triggering application event, as described above, and to store the performance data in one or more data records. Each data record, for example, may include a collection of field-value pairs, each field-value pair storing a particular item of performance data in association with a field for the item. For example, a data record generated by a monitoring component **112** may include a "networkLatency" field (not shown in the Figure) in which a value is stored. This field indicates a network latency measurement associated with one or more network requests. The data record may include a "state" field to store a value indicating a state of a network connection, and so forth for any number of aspects of collected performance data.

[0086] 2.4. Data Server System

[0087] FIG. **2** depicts a block diagram of an exemplary data intake and query system **108**, similar to the SPLUNK® ENTERPRISE system. System **108** includes one or more forwarders **204** that receive data from a variety of input data sources **202**, and one or more indexers **206** that process and store the data in one or more data stores **208**. These forwarders and indexers can comprise separate computer systems, or may alternatively comprise separate processes executing on one or more computer systems.

[0088] Each data source **202** broadly represents a distinct source of data that can be consumed by a system **108**. Examples of a data source **202** include, without limitation, data files, directories of files, data sent over a network, event logs, registries, etc.

[0089] During operation, the forwarders **204** identify which indexers **206** receive data collected from a data source **202** and forward the data to the appropriate indexers. Forwarders **204** can also perform operations on the data before forwarding, including removing extraneous data, detecting timestamps in the data, parsing data, indexing data, routing data based on criteria relating to the data being routed, and/or performing other data transformations.

[0090] In an embodiment, a forwarder **204** may comprise a service accessible to client devices **102** and host devices **106** via a network **104**. For example, one type of forwarder **204** may be capable of consuming vast amounts of real-time data from a potentially large number of client devices **102** and/or host devices **106**. The forwarder **204** may, for example, comprise a computing device which implements multiple data pipelines or "queues" to handle forwarding of network data to indexers **206**. A forwarder **204** may also perform many of the functions that are performed by an indexer. For example, a forwarder **204** may perform keyword extractions on raw data or parse raw data to create events. A forwarder **204** may generate time stamps for events. Additionally or alternatively, a forwarder **204** may perform routing of events to indexers. Data store **208** may contain events derived from machine data from a variety of sources all pertaining to the same component in an IT environment, and this data may be produced by the machine in question or by other components in the IT environment.

[0091] 2.5. Data Ingestion

[0092] FIG. **3** depicts a flow chart illustrating an example data flow performed by Data Intake and Query system **108**, in accordance with the disclosed embodiments. The data flow illustrated in FIG. **3** is provided for illustrative purposes only; those skilled in the art would understand that one or more of the steps of the processes illustrated in FIG. **3** may be removed or the ordering of the steps may be changed. Furthermore, for the purposes of illustrating a clear example, one or more particular system components are described in the context of performing various operations during each of the data flow stages. For example, a forwarder is described as receiving and processing data during an input phase; an indexer is described as parsing and indexing data during parsing and indexing phases; and a search head is described as performing a search query during a search phase. How-

ever, other system arrangements and distributions of the processing steps across system components may be used.

[0093] 2.5.1. Input

[0094] At block **302**, a forwarder receives data from an input source, such as a data source **202** shown in FIG. **2**. A forwarder initially may receive the data as a raw data stream generated by the input source. For example, a forwarder may receive a data stream from a log file generated by an application server, from a stream of network data from a network device, or from any other source of data. In one embodiment, a forwarder receives the raw data and may segment the data stream into "blocks", or "buckets," possibly of a uniform data size, to facilitate subsequent processing steps.

[0095] At block **304**, a forwarder or other system component annotates each block generated from the raw data with one or more metadata fields. These metadata fields may, for example, provide information related to the data block as a whole and may apply to each event that is subsequently derived from the data in the data block. For example, the metadata fields may include separate fields specifying each of a host, a source, and a source type related to the data block. A host field may contain a value identifying a host name or IP address of a device that generated the data. A source field may contain a value identifying a source of the data, such as a pathname of a file or a protocol and port related to received network data. A source type field may contain a value specifying a particular source type label for the data. Additional metadata fields may also be included during the input phase, such as a character encoding of the data, if known, and possibly other values that provide information relevant to later processing steps. In an embodiment, a forwarder forwards the annotated data blocks to another system component (typically an indexer) for further processing.

[0096] The SPLUNK® ENTERPRISE system allows forwarding of data from one SPLUNK® ENTERPRISE instance to another, or even to a third-party system. SPLUNK® ENTERPRISE system can employ different types of forwarders in a configuration.

[0097] In an embodiment, a forwarder may contain the essential components needed to forward data. It can gather data from a variety of inputs and forward the data to a SPLUNK® ENTERPRISE server for indexing and searching. It also can tag metadata (e.g., source, source type, host, etc.).

[0098] Additionally or optionally, in an embodiment, a forwarder has the capabilities of the aforementioned forwarder as well as additional capabilities. The forwarder can parse data before forwarding the data (e.g., associate a time stamp with a portion of data and create an event, etc.) and can route data based on criteria such as source or type of event. It can also index data locally while forwarding the data to another indexer.

[0099] 2.5.2. Parsing

[0100] At block **306**, an indexer receives data blocks from a forwarder and parses the data to organize the data into events. In an embodiment, to organize the data into events, an indexer may determine a source type associated with each data block (e.g., by extracting a source type label from the metadata fields associated with the data block, etc.) and refer to a source type configuration corresponding to the identified source type. The source type definition may include one or more properties that indicate to the indexer to automatically determine the boundaries of events within the data. In general, these properties may include regular expression-based rules or delimiter rules where, for example, event boundaries may be indicated by predefined characters or character strings. These predefined characters may include punctuation marks or other special characters including, for example, carriage returns, tabs, spaces, line breaks, etc. If a source type for the data is unknown to the indexer, an indexer may infer a source type for the data by examining the structure of the data. Then, it can apply an inferred source type definition to the data to create the events.

[0101] At block **308**, the indexer determines a timestamp for each event. Similar to the process for creating events, an indexer may again refer to a source type definition associated with the data to locate one or more properties that indicate instructions for determining a timestamp for each event. The properties may, for example, instruct an indexer to extract a time value from a portion of data in the event, to interpolate time values based on timestamps associated with temporally proximate events, to create a timestamp based on a time the event data was received or generated, to use the timestamp of a previous event, or use any other rules for determining timestamps.

[0102] At block **310**, the indexer associates with each event one or more metadata fields including a field containing the timestamp (in some embodiments, a timestamp may be included in the metadata fields) determined for the event. These metadata fields may include a number of "default fields" that are associated with all events, and may also include one more custom fields as defined by a user. Similar to the metadata fields associated with the data blocks at block **304**, the default metadata fields associated with each event may include a host, source, and source type field including or in addition to a field storing the timestamp.

[0103] At block **312**, an indexer may optionally apply one or more transformations to data included in the events created at block **306**. For example, such transformations can include removing a portion of an event (e.g., a portion used to define event boundaries, extraneous characters from the event, other extraneous text, etc.), masking a portion of an event (e.g., masking a credit card number), removing redundant portions of an event, etc. The transformations applied to event data may, for example, be specified in one or more configuration files and referenced by one or more source type definitions.

[0104] 2.5.3. Indexing

[0105] At blocks **314** and **316**, an indexer can optionally generate a keyword index to facilitate fast keyword searching for event data. To build a keyword index, at block **314**, the indexer identifies a set of keywords in each event. At block **316**, the indexer includes the identified keywords in an index, which associates each stored keyword with reference pointers to events containing that keyword (or to locations within events where that keyword is located, other location identifiers, etc.). When an indexer subsequently receives a keyword-based query, the indexer can access the keyword index to quickly identify events containing the keyword.

[0106] In some embodiments, the keyword index may include entries for name-value pairs found in events, where a name-value pair can include a pair of keywords connected by a symbol, such as an equals sign or colon. This way, events containing these name-value pairs can be quickly located. In some embodiments, fields can automatically be generated for some or all of the name-value pairs at the time

of indexing. For example, if the string "dest=10.0.1.2" is found in an event, a field named "dest" may be created for the event, and assigned a value of "10.0.1.2".

[0107] At block **318**, the indexer stores the events with an associated timestamp in a data store **208**. Timestamps enable a user to search for events based on a time range. In one embodiment, the stored events are organized into "buckets," where each bucket stores events associated with a specific time range based on the timestamps associated with each event. This may not only improve time-based searching, but also allows for events with recent timestamps, which may have a higher likelihood of being accessed, to be stored in a faster memory to facilitate faster retrieval. For example, buckets containing the most recent events can be stored in flash memory rather than on a hard disk.

[0108] Each indexer **206** may be responsible for storing and searching a subset of the events contained in a corresponding data store **208**. By distributing events among the indexers and data stores, the indexers can analyze events for a query in parallel. For example, using map-reduce techniques, each indexer returns partial responses for a subset of events to a search head that combines the results to produce an answer for the query. By storing events in buckets for specific time ranges, an indexer may further optimize data retrieval process by searching buckets corresponding to time ranges that are relevant to a query.

[0109] Moreover, events and buckets can also be replicated across different indexers and data stores to facilitate high availability and disaster recovery as described in U.S. patent application Ser. No. 14/266,812, entitled "Site-Based Search Affinity", filed on 30 Apr. 2014, and in U.S. patent application Ser. No. 14/266,817, entitled "Multi-Site Clustering", also filed on 30 Apr. 2014, each of which is hereby incorporated by reference in its entirety for all purposes.

[0110] 2.6. Query Processing

[0111] FIG. **4** is a flow diagram that illustrates an example process that a search head and one or more indexers may perform during a search query. At block **402**, a search head receives a search query from a client. At block **404**, the search head analyzes the search query to determine what portion(s) of the query can be delegated to indexers and what portions of the query can be executed locally by the search head. At block **406**, the search head distributes the determined portions of the query to the appropriate indexers. In an embodiment, a search head cluster may take the place of an independent search head where each search head in the search head cluster coordinates with peer search heads in the search head cluster to schedule jobs, replicate search results, update configurations, fulfill search requests, etc. In an embodiment, the search head (or each search head) communicates with a master node (also known as a cluster master, not shown in Fig.) that provides the search head with a list of indexers to which the search head can distribute the determined portions of the query. The master node maintains a list of active indexers and can also designate which indexers may have responsibility for responding to queries over certain sets of events. A search head may communicate with the master node before the search head distributes queries to indexers to discover the addresses of active indexers.

[0112] At block **408**, the indexers to which the query was distributed, search data stores associated with them for events that are responsive to the query. To determine which events are responsive to the query, the indexer searches for events that match the criteria specified in the query. These criteria can include matching keywords or specific values for certain fields. The searching operations at block **408** may use the late-binding schema to extract values for specified fields from events at the time the query is processed. In an embodiment, one or more rules for extracting field values may be specified as part of a source type definition. The indexers may then either send the relevant events back to the search head, or use the events to determine a partial result, and send the partial result back to the search head.

[0113] At block **410**, the search head combines the partial results and/or events received from the indexers to produce a final result for the query. This final result may comprise different types of data depending on what the query requested. For example, the results can include a listing of matching events returned by the query, or some type of visualization of the data from the returned events. In another example, the final result can include one or more calculated values derived from the matching events.

[0114] The results generated by the system **108** can be returned to a client using different techniques. For example, one technique streams results or relevant events back to a client in real-time as they are identified. Another technique waits to report the results to the client until a complete set of results (which may include a set of relevant events or a result based on relevant events) is ready to return to the client. Yet another technique streams interim results or relevant events back to the client in real-time until a complete set of results is ready, and then returns the complete set of results to the client. In another technique, certain results are stored as "search jobs" and the client may retrieve the results by referring the search jobs.

[0115] The search head can also perform various operations to make the search more efficient. For example, before the search head begins execution of a query, the search head can determine a time range for the query and a set of common keywords that all matching events include. The search head may then use these parameters to query the indexers to obtain a superset of the eventual results. Then, during a filtering stage, the search head can perform field-extraction operations on the superset to produce a reduced set of search results. This speeds up queries that are performed on a periodic basis.

[0116] 2.7. Field Extraction

[0117] The search head **210** allows users to search and visualize event data extracted from raw machine data received from homogenous data sources. It also allows users to search and visualize event data extracted from raw machine data received from heterogeneous data sources. The search head **210** includes various mechanisms, which may additionally reside in an indexer **206**, for processing a query. Splunk Processing Language (SPL), used in conjunction with the SPLUNK® ENTERPRISE system, can be utilized to make a query. SPL is a pipelined search language in which a set of inputs is operated on by a first command in a command line, and then a subsequent command following the pipe symbol "I" operates on the results produced by the first command, and so on for additional commands. Other query languages, such as the Structured Query Language ("SQL"), can be used to create a query.

[0118] In response to receiving the search query, search head **210** uses extraction rules to extract values for the fields associated with a field or fields in the event data being searched. The search head **210** obtains extraction rules that

specify how to extract a value for certain fields from an event. Extraction rules can comprise regex rules that specify how to extract values for the relevant fields. In addition to specifying how to extract field values, the extraction rules may also include instructions for deriving a field value by performing a function on a character string or value retrieved by the extraction rule. For example, a transformation rule may truncate a character string, or convert the character string into a different data format. In some cases, the query itself can specify one or more extraction rules.

[0119] The search head **210** can apply the extraction rules to event data that it receives from indexers **206**. Indexers **206** may apply the extraction rules to events in an associated data store **208**. Extraction rules can be applied to all the events in a data store, or to a subset of the events that have been filtered based on some criteria (e.g., event time stamp values, etc.). Extraction rules can be used to extract one or more values for a field from events by parsing the event data and examining the event data for one or more patterns of characters, numbers, delimiters, etc., that indicate where the field begins and, optionally, ends.

[0120] FIG. **5** illustrates an example of raw machine data received from disparate data sources. In this example, a firewall **501** running on the network processes traffic from outside the company and allows or denies traffic. The firewall **501** creates a log entry **504** for the traffic processed. A router **502** running on the network routes traffic through the network. The router **502** creates a log entry **505** for the traffic processed. The systems **501** and **502** are disparate systems that do not have a common logging format. The log data **504** and **505** are sent to the SPLUNK® ENTERPRISE system in different formats.

[0121] Using the log data received at one or more indexers **206** from the systems, the vendor can uniquely obtain an insight into device behavior. The search head **210** allows the vendor's administrator to search the log data from the systems that one or more indexers **206** are responsible for searching, thereby obtaining correlated information, such as the device ID. The device ID field value exists in the data gathered from the systems, but the device ID field value may be located in different areas of the data given differences in the architecture of the systems—there is a semantic relationship between the device ID field values generated by the systems. The search head **210** requests event data from the one or more indexers **206** to gather relevant event data from the systems. It then applies extraction rules to the event data in order to extract field values that it can correlate. The search head may apply a different extraction rule to each set of events from each system when the event data format differs among systems.

[0122] Note that query results can be returned to a client, a search head, or any other system component for further processing. In general, query results may include a set of one or more events, a set of one or more values obtained from the events, a subset of the values, statistics calculated based on the values, a report containing the values, or a visualization, such as a graph or chart, generated from the values.

[0123] 2.8. Data Models

[0124] A data model is a hierarchically structured search-time mapping of semantic knowledge about one or more datasets. It encodes the domain knowledge necessary to build a variety of specialized searches of those datasets. Those searches, in turn, can be used to generate reports.

[0125] A data model is composed of one or more "objects" (or "data model objects") that define or otherwise correspond to a specific set of data.

[0126] Objects in data models can be arranged hierarchically in parent/child relationships. Each child object represents a subset of the dataset covered by its parent object. The top-level objects in data models are collectively referred to as "root objects."

[0127] Child objects have inheritance. Data model objects are defined by characteristics that mostly break down into constraints and attributes. Child objects inherit constraints and attributes from their parent objects and have additional constraints and attributes of their own. Child objects provide a way of filtering events from parent objects. Because a child object always provides an additional constraint in addition to the constraints it has inherited from its parent object, the dataset it represents is always a subset of the dataset that its parent represents.

[0128] For example, a first data model object may define a broad set of data pertaining to e-mail activity generally, and another data model object may define specific datasets within the broad dataset, such as a subset of the e-mail data pertaining specifically to e-mails sent. Examples of data models can include electronic mail, authentication, databases, intrusion detection, malware, application state, alerts, compute inventory, network sessions, network traffic, performance, audits, updates, vulnerabilities, etc. Data models and their objects can be designed by knowledge managers in an organization, and they can enable downstream users to quickly focus on a specific set of data. For example, a user can simply select an "e-mail activity" data model object to access a dataset relating to e-mails generally (e.g., sent or received), or select an "e-mails sent" data model object (or data sub-model object) to access a dataset relating to e-mails sent.

[0129] A data model object may be defined by (1) a set of search constraints, and (2) a set of fields. Thus, a data model object can be used to quickly search data to identify a set of events and to identify a set of fields to be associated with the set of events. For example, an "e-mails sent" data model object may specify a search for events relating to e-mails that have been sent, and specify a set of fields that are associated with the events. Thus, a user can retrieve and use the "e-mails sent" data model object to quickly search source data for events relating to sent e-mails, and may be provided with a listing of the set of fields relevant to the events in a user interface screen.

[0130] A child of the parent data model may be defined by a search (typically a narrower search) that produces a subset of the events that would be produced by the parent data model's search. The child's set of fields can include a subset of the set of fields of the parent data model and/or additional fields. Data model objects that reference the subsets can be arranged in a hierarchical manner, so that child subsets of events are proper subsets of their parents. A user iteratively applies a model development tool (not shown in Fig.) to prepare a query that defines a subset of events and assigns an object name to that subset. A child subset is created by further limiting a query that generated a parent subset. A late-binding schema of field extraction rules is associated with each object or subset in the data model.

[0131] Data definitions in associated schemas can be taken from the common information model (CIM) or can be devised for a particular schema and optionally added to the

CIM. Child objects inherit fields from parents and can include fields not present in parents. A model developer can select fewer extraction rules than are available for the sources returned by the query that defines events belonging to a model. Selecting a limited set of extraction rules can be a tool for simplifying and focusing the data model, while allowing a user flexibility to explore the data subset. Development of a data model is further explained in U.S. Pat. Nos. 8,788,525 and 8,788,526, both entitled "DATA MODEL FOR MACHINE DATA FOR SEMANTIC SEARCH", both issued on 22 Jul. 2014, U.S. Pat. No. 8,983,994, entitled "GENERATION OF A DATA MODEL FOR SEARCHING MACHINE DATA", issued on 17 Mar. 2015, U.S. patent application Ser. No. 14/611,232, entitled "GENERATION OF A DATA MODEL APPLIED TO QUERIES", filed on 31 Jan. 2015, and U.S. patent application Ser. No. 14/815,884, entitled "GENERATION OF A DATA MODEL APPLIED TO OBJECT QUERIES", filed on 31 Jul. 2015, each of which is hereby incorporated by reference in its entirety for all purposes. See, also, Knowledge Manager Manual, Build a Data Model, Splunk Enterprise 6.1.3 pp. 150-204 (Aug. 25, 2014).

[0132] A data model can also include reports. One or more report formats can be associated with a particular data model and be made available to run against the data model. A user can use child objects to design reports with object datasets that already have extraneous data pre-filtered out. In an embodiment, the data intake and query system **108** provides the user with the ability to produce reports (e.g., a table, chart, visualization, etc.) without having to enter SPL, SQL, or other query language terms into a search screen. Data models are used as the basis for the search feature.

[0133] Data models may be selected in a report generation interface. The report generator supports drag-and-drop organization of fields to be summarized in a report. When a model is selected, the fields with available extraction rules are made available for use in the report. The user may refine and/or filter search results to produce more precise reports. The user may select some fields for organizing the report and select other fields for providing detail according to the report organization. For example, "region" and "salesperson" are fields used for organizing the report and sales data can be summarized (subtotaled and totaled) within this organization. The report generator allows the user to specify one or more fields within events and apply statistical analysis on values extracted from the specified one or more fields. The report generator may aggregate search results across sets of events and generate statistics based on aggregated search results. Building reports using the report generation interface is further explained in U.S. patent application Ser. No. 14/503,335, entitled "GENERATING REPORTS FROM UNSTRUCTURED DATA", filed on 30 Sep. 2014, and which is hereby incorporated by reference in its entirety for all purposes, and in Pivot Manual, Splunk Enterprise 6.1.3 (Aug. 4, 2014). Data visualizations also can be generated in a variety of formats, by reference to the data model. Reports, data visualizations, and data model objects can be saved and associated with the data model for future use. The data model object may be used to perform searches of other data.

[0134] 2.9. Acceleration Technique

[0135] The above-described system provides significant flexibility by enabling analysis of massive quantities of minimally processed data "on the fly" at search time instead of storing pre-specified portions of the data in a database at ingestion time. This flexibility enables valuable insights, data correlation, and the performance of subsequent queries to examine interesting aspects of the data that may not have been apparent at ingestion time.

[0136] However, performing extraction and analysis operations at search time can involve a large amount of data and require a large number of computational operations, which can cause delays in processing the queries. Advantageously, SPLUNK® ENTERPRISE system employs a number of unique acceleration techniques that have been developed to speed up analysis operations performed at search time. These techniques include: (1) performing search operations in parallel across multiple indexers; (2) using a keyword index; (3) using a high performance analytics store; and (4) accelerating the process of generating reports. These novel techniques are described in more detail below.

[0137] 2.9.1. Aggregation Technique

[0138] To facilitate faster query processing, a query can be structured such that multiple indexers perform the query in parallel, while aggregation of search results from the multiple indexers is performed locally at the search head. For example, FIG. **6** illustrates how a search query **602** received from a client at a search head **210** can split into two phases, including: (1) subtasks **604** (e.g., data retrieval or simple filtering) that may be performed in parallel by indexers **206** for execution, and (2) a search results aggregation operation **606** to be executed by the search head when the results are ultimately collected from the indexers.

[0139] During operation, upon receiving search query **602**, a search head **210** determines that a portion of the operations involved with the search query may be performed locally by the search head. The search head modifies search query **602** by substituting "stats" (create aggregate statistics over results sets received from the indexers at the search head) with "prestats" (create statistics by the indexer from local results set) to produce search query **604**, and then distributes search query **604** to distributed indexers, which are also referred to as "search peers." Note that search queries may generally specify search criteria or operations to be performed on events that meet the search criteria. Search queries may also specify field names, as well as search criteria for the values in the fields or operations to be performed on the values in the fields. Moreover, the search head may distribute the full search query to the search peers as illustrated in FIG. **4**, or may alternatively distribute a modified version (e.g., a more restricted version) of the search query to the search peers. In this example, the indexers are responsible for producing the results and sending them to the search head. After the indexers return the results to the search head, the search head aggregates the received results **606** to form a single search result set. By executing the query in this manner, the system effectively distributes the computational operations across the indexers while minimizing data transfers.

[0140] 2.9.2. Keyword Index

[0141] As described above with reference to the flow charts in FIG. **3** and FIG. **4**, data intake and query system **108** can construct and maintain one or more keyword indices to quickly identify events containing specific keywords. This technique can greatly speed up the processing of queries involving specific keywords. As mentioned above, to build a keyword index, an indexer first identifies a set of keywords. Then, the indexer includes the identified keywords in an index, which associates each stored keyword

with references to events containing that keyword, or to locations within events where that keyword is located. When an indexer subsequently receives a keyword-based query, the indexer can access the keyword index to quickly identify events containing the keyword.

[0142] 2.9.3. High Performance Analytics Store

[0143] To speed up certain types of queries, some embodiments of system **108** create a high performance analytics store, which is referred to as a "summarization table," that contains entries for specific field-value pairs. Each of these entries keeps track of instances of a specific value in a specific field in the event data and includes references to events containing the specific value in the specific field. For example, an example entry in a summarization table can keep track of occurrences of the value "94107" in a "ZIP code" field of a set of events and the entry includes references to all of the events that contain the value "94107" in the ZIP code field. This optimization technique enables the system to quickly process queries that seek to determine how many events have a particular value for a particular field. To this end, the system can examine the entry in the summarization table to count instances of the specific value in the field without having to go through the individual events or perform data extractions at search time. Also, if the system needs to process all events that have a specific field-value combination, the system can use the references in the summarization table entry to directly access the events to extract further information without having to search all of the events to find the specific field-value combination at search time.

[0144] In some embodiments, the system maintains a separate summarization table for each of the above-described time-specific buckets that stores events for a specific time range. A bucket-specific summarization table includes entries for specific field-value combinations that occur in events in the specific bucket. Alternatively, the system can maintain a separate summarization table for each indexer. The indexer-specific summarization table includes entries for the events in a data store that are managed by the specific indexer. Indexer-specific summarization tables may also be bucket-specific.

[0145] The summarization table can be populated by running a periodic query that scans a set of events to find instances of a specific field-value combination, or alternatively instances of all field-value combinations for a specific field. A periodic query can be initiated by a user, or can be scheduled to occur automatically at specific time intervals. A periodic query can also be automatically launched in response to a query that asks for a specific field-value combination.

[0146] In some cases, when the summarization tables may not cover all of the events that are relevant to a query, the system can use the summarization tables to obtain partial results for the events that are covered by summarization tables, but may also have to search through other events that are not covered by the summarization tables to produce additional results. These additional results can then be combined with the partial results to produce a final set of results for the query. The summarization table and associated techniques are described in more detail in U.S. Pat. No. 8,682,925, entitled "Distributed High Performance Analytics Store", issued on 25 Mar. 2014, U.S. patent application Ser. No. 14/170,159, entitled "SUPPLEMENTING A HIGH PERFORMANCE ANALYTICS STORE WITH EVALUATION OF INDIVIDUAL EVENTS TO RESPOND TO AN EVENT QUERY", filed on 31 Jan. 2014, and U.S. patent application Ser. No. 14/815,973, entitled "STORAGE MEDIUM AND CONTROL DEVICE", filed on 21 Feb. 2014, each of which is hereby incorporated by reference in its entirety.

[0147] 2.9.4. Accelerating Report Generation

[0148] In some embodiments, a data server system such as the SPLUNK® ENTERPRISE system can accelerate the process of periodically generating updated reports based on query results. To accelerate this process, a summarization engine automatically examines the query to determine whether generation of updated reports can be accelerated by creating intermediate summaries. If reports can be accelerated, the summarization engine periodically generates a summary covering data obtained during a latest non-overlapping time period. For example, where the query seeks events meeting a specified criterion, a summary for the time period includes only events within the time period that meet the specified criteria. Similarly, if the query seeks statistics calculated from the events, such as the number of events that match the specified criteria, then the summary for the time period includes the number of events in the period that match the specified criteria.

[0149] In addition to the creation of the summaries, the summarization engine schedules the periodic updating of the report associated with the query. During each scheduled report update, the query engine determines whether intermediate summaries have been generated covering portions of the time period covered by the report update. If so, then the report is generated based on the information contained in the summaries. Also, if additional event data has been received and has not yet been summarized, and is required to generate the complete report, the query can be run on this additional event data. Then, the results returned by this query on the additional event data, along with the partial results obtained from the intermediate summaries, can be combined to generate the updated report. This process is repeated each time the report is updated. Alternatively, if the system stores events in buckets covering specific time ranges, then the summaries can be generated on a bucket-by-bucket basis. Note that producing intermediate summaries can save the work involved in re-running the query for previous time periods, so advantageously only the newer event data needs to be processed while generating an updated report. These report acceleration techniques are described in more detail in U.S. Pat. No. 8,589,403, entitled "Compressed Journaling In Event Tracking Files For Metadata Recovery And Replication", issued on 19 Nov. 2013, U.S. Pat. No. 8,412,696, entitled "Real Time Searching And Reporting", issued on 2 Apr. 2011, and U.S. Pat. Nos. 8,589,375 and 8,589,432, both also entitled "REAL TIME SEARCHING AND REPORTING", both issued on 19 Nov. 2013, each of which is hereby incorporated by reference in its entirety.

[0150] 2.10. Security Features

[0151] The SPLUNK® ENTERPRISE platform provides various schemas, dashboards and visualizations that simplify developers' task to create applications with additional capabilities. One such application is the SPLUNK® APP FOR ENTERPRISE SECURITY, which performs monitoring and alerting operations and includes analytics to facilitate identifying both known and unknown security threats based on large volumes of data stored by the SPLUNK®

ENTERPRISE system. SPLUNK® APP FOR ENTERPRISE SECURITY provides the security practitioner with visibility into security-relevant threats found in the enterprise infrastructure by capturing, monitoring, and reporting on data from enterprise security devices, systems, and applications. Through the use of SPLUNK® ENTERPRISE searching and reporting capabilities, SPLUNK® APP FOR ENTERPRISE SECURITY provides a top-down and bottom-up view of an organization's security posture.

[0152] The SPLUNK® APP FOR ENTERPRISE SECURITY leverages SPLUNK® ENTERPRISE search-time normalization techniques, saved searches, and correlation searches to provide visibility into security-relevant threats and activity and generate notable events for tracking. The App enables the security practitioner to investigate and explore the data to find new or unknown threats that do not follow signature-based patterns.

[0153] Conventional Security Information and Event Management (SIEM) systems that lack the infrastructure to effectively store and analyze large volumes of security-related data. Traditional SIEM systems typically use fixed schemas to extract data from pre-defined security-related fields at data ingestion time and storing the extracted data in a relational database. This traditional data extraction process (and associated reduction in data size) that occurs at data ingestion time inevitably hampers future incident investigations that may need original data to determine the root cause of a security issue, or to detect the onset of an impending security threat.

[0154] In contrast, the SPLUNK® APP FOR ENTERPRISE SECURITY system stores large volumes of minimally processed security-related data at ingestion time for later retrieval and analysis at search time when a live security threat is being investigated. To facilitate this data retrieval process, the SPLUNK® APP FOR ENTERPRISE SECURITY provides pre-specified schemas for extracting relevant values from the different types of security-related event data and enables a user to define such schemas.

[0155] The SPLUNK® APP FOR ENTERPRISE SECURITY can process many types of security-related information. In general, this security-related information can include any information that can be used to identify security threats. For example, the security-related information can include network-related information, such as IP addresses, domain names, asset identifiers, network traffic volume, uniform resource locator strings, and source addresses. The process of detecting security threats for network-related information is further described in U.S. Pat. No. 8,826,434, entitled "SECURITY THREAT DETECTION BASED ON INDICATIONS IN BIG DATA OF ACCESS TO NEWLY REGISTERED DOMAINS", issued on 2 Sep. 2014, U.S. patent application Ser. No. 13/956,252, entitled "INVESTIGATIVE AND DYNAMIC DETECTION OF POTENTIAL SECURITY-THREAT INDICATORS FROM EVENTS IN BIG DATA", filed on 31 Jul. 2013, U.S. patent application Ser. No. 14/445,018, entitled "GRAPHIC DISPLAY OF SECURITY THREATS BASED ON INDICATIONS OF ACCESS TO NEWLY REGISTERED DOMAINS", filed on 28 Jul. 2014, U.S. patent application Ser. No. 14/445,023, entitled "SECURITY THREAT DETECTION OF NEWLY REGISTERED DOMAINS", filed on 28 Jul. 2014, U.S. patent application Ser. No. 14/815,971, entitled "SECURITY THREAT DETECTION USING DOMAIN NAME ACCESSES", filed on 1 Aug. 2015, and U.S. patent application Ser. No. 14/815,972, entitled "SECURITY THREAT DETECTION USING DOMAIN NAME REGISTRATIONS", filed on 1 Aug. 2015, each of which is hereby incorporated by reference in its entirety for all purposes. Security-related information can also include malware infection data and system configuration information, as well as access control information, such as login/logout information and access failure notifications. The security-related information can originate from various sources within a data center, such as hosts, virtual machines, storage devices and sensors. The security-related information can also originate from various sources in a network, such as routers, switches, email servers, proxy servers, gateways, firewalls and intrusion-detection systems.

[0156] During operation, the SPLUNK® APP FOR ENTERPRISE SECURITY facilitates detecting "notable events" that are likely to indicate a security threat. These notable events can be detected in a number of ways: (1) a user can notice a correlation in the data and can manually identify a corresponding group of one or more events as "notable;" or (2) a user can define a "correlation search" specifying criteria for a notable event, and every time one or more events satisfy the criteria, the application can indicate that the one or more events are notable. A user can alternatively select a pre-defined correlation search provided by the application. Note that correlation searches can be run continuously or at regular intervals (e.g., every hour) to search for notable events. Upon detection, notable events can be stored in a dedicated "notable events index," which can be subsequently accessed to generate various visualizations containing security-related information. Also, alerts can be generated to notify system operators when important notable events are discovered.

[0157] 2.11. Data Center Monitoring

[0158] As mentioned above, the SPLUNK® ENTERPRISE platform provides various features that simplify the developer's task to create various applications. One such application is SPLUNK® APP FOR VMWARE® that provides operational visibility into granular performance metrics, logs, tasks and events, and topology from hosts, virtual machines and virtual centers. It empowers administrators with an accurate real-time picture of the health of the environment, proactively identifying performance and capacity bottlenecks.

[0159] Conventional data-center-monitoring systems lack the infrastructure to effectively store and analyze large volumes of machine-generated data, such as performance information and log data obtained from the data center. In conventional data-center-monitoring systems, machine-generated data is typically pre-processed prior to being stored, for example, by extracting pre-specified data items and storing them in a database to facilitate subsequent retrieval and analysis at search time. However, the rest of the data is not saved and discarded during pre-processing.

[0160] In contrast, the SPLUNK® APP FOR VMWARE® stores large volumes of minimally processed machine data, such as performance information and log data, at ingestion time for later retrieval and analysis at search time when a live performance issue is being investigated. In addition to data obtained from various log files, this performance-related information can include values for performance metrics obtained through an application programming interface (API) provided as part of the vSphere Hypervisor™ system distributed by VMware, Inc. of Palo Alto, Calif. For

example, these performance metrics can include: (1) CPU-related performance metrics; (2) disk-related performance metrics; (3) memory-related performance metrics; (4) network-related performance metrics; (5) energy-usage statistics; (6) data-traffic-related performance metrics; (7) overall system availability performance metrics; (8) cluster-related performance metrics; and (9) virtual machine performance statistics. Such performance metrics are described in U.S. patent application Ser. No. 14/167,316, entitled "Correlation For User-Selected Time Ranges Of Values For Performance Metrics Of Components In An Information-Technology Environment With Log Data From That Information-Technology Environment", filed on 29 Jan. 2014, and which is hereby incorporated by reference in its entirety for all purposes.

[0161] To facilitate retrieving information of interest from performance data and log files, the SPLUNK® APP FOR VMWARE® provides pre-specified schemas for extracting relevant values from different types of performance-related event data, and also enables a user to define such schemas.

[0162] 2.12. Cloud-Based System Overview

[0163] The example data intake and query system **108** described in reference to FIG. **2** comprises several system components, including one or more forwarders, indexers, and search heads. In some environments, a user of a data intake and query system **108** may install and configure, on computing devices owned and operated by the user, one or more software applications that implement some or all of these system components. For example, a user may install a software application on server computers owned by the user and configure each server to operate as one or more of a forwarder, an indexer, a search head, etc. This arrangement generally may be referred to as an "on-premises" solution. That is, the system **108** is installed and operates on computing devices directly controlled by the user of the system. Some users may prefer an on-premises solution because it may provide a greater level of control over the configuration of certain aspects of the system (e.g., security, privacy, standards, controls, etc.). However, other users may instead prefer an arrangement in which the user is not directly responsible for providing and managing the computing devices upon which various components of system **108** operate.

[0164] In one embodiment, to provide an alternative to an entirely on-premises environment for system **108**, one or more of the components of a data intake and query system instead may be provided as a cloud-based service. In this context, a cloud-based service refers to a service hosted by one more computing resources that are accessible to end users over a network, for example, by using a web browser or other application on a client device to interface with the remote computing resources. For example, a service provider may provide a cloud-based data intake and query system by managing computing resources configured to implement various aspects of the system (e.g., forwarders, indexers, search heads, etc.) and by providing access to the system to end users via a network. Typically, a user may pay a subscription or other fee to use such a service. Each subscribing user of the cloud-based service may be provided with an account that enables the user to configure a customized cloud-based system based on the user's preferences.

[0165] FIG. **7** illustrates a block diagram of an example cloud-based data intake and query system. Similar to the system of FIG. **2**, the networked computer system **700** includes input data sources **202** and forwarders **204**. These input data sources and forwarders may be in a subscriber's private computing environment. Alternatively, they might be directly managed by the service provider as part of the cloud service. In the example system **700**, one or more forwarders **204** and client devices **702** are coupled to a cloud-based data intake and query system **706** via one or more networks **704**. Network **704** broadly represents one or more LANs, WANs, cellular networks, intranetworks, internetworks, etc., using any of wired, wireless, terrestrial microwave, satellite links, etc., and may include the public Internet, and is used by client devices **702** and forwarders **204** to access the system **706**. Similar to the system of **108**, each of the forwarders **204** may be configured to receive data from an input source and to forward the data to other components of the system **706** for further processing.

[0166] In an embodiment, a cloud-based data intake and query system **706** may comprise a plurality of system instances **708**. In general, each system instance **708** may include one or more computing resources managed by a provider of the cloud-based system **706** made available to a particular subscriber. The computing resources comprising a system instance **708** may, for example, include one or more servers or other devices configured to implement one or more forwarders, indexers, search heads, and other components of a data intake and query system, similar to system **108**. As indicated above, a subscriber may use a web browser or other application of a client device **702** to access a web portal or other interface that enables the subscriber to configure an instance **708**.

[0167] Providing a data intake and query system as described in reference to system **108** as a cloud-based service presents a number of challenges. Each of the components of a system **108** (e.g., forwarders, indexers and search heads) may at times refer to various configuration files stored locally at each component. These configuration files typically may involve some level of user configuration to accommodate particular types of data a user desires to analyze and to account for other user preferences. However, in a cloud-based service context, users typically may not have direct access to the underlying computing resources implementing the various system components (e.g., the computing resources comprising each system instance **708**) and may desire to make such configurations indirectly, for example, using one or more web-based interfaces. Thus, the techniques and systems described herein for providing user interfaces that enable a user to configure source type definitions are applicable to both on-premises and cloud-based service contexts, or some combination thereof (e.g., a hybrid system where both an on-premises environment such as SPLUNK® ENTERPRISE and a cloud-based environment such as SPLUNK CLOUD™ are centrally visible).

[0168] 2.13. Searching Externally Archived Data

[0169] FIG. **8** shows a block diagram of an example of a data intake and query system **108** that provides transparent search facilities for data systems that are external to the data intake and query system. Such facilities are available in the HUNK® system provided by Splunk Inc. of San Francisco, Calif. HUNK® represents an analytics platform that enables business and IT teams to rapidly explore, analyze, and visualize data in Hadoop and NoSQL data stores.

[0170] The search head **210** of the data intake and query system receives search requests from one or more client devices **804** over network connections **820**. As discussed

above, the data intake and query system **108** may reside in an enterprise location, in the cloud, etc. FIG. **8** illustrates that multiple client devices **804***a*, **804***b*, . . . , **804***n* may communicate with the data intake and query system **108**. The client devices **804** may communicate with the data intake and query system using a variety of connections. For example, one client device in FIG. **8** is illustrated as communicating over an Internet (Web) protocol, another client device is illustrated as communicating via a command line interface, and another client device is illustrated as communicating via a system developer kit (SDK).

[0171] The search head **210** analyzes the received search request to identify request parameters. If a search request received from one of the client devices **804** references an index maintained by the data intake and query system, then the search head **210** connects to one or more indexers **206** of the data intake and query system for the index referenced in the request parameters. That is, if the request parameters of the search request reference an index, then the search head accesses the data in the index via the indexer. The data intake and query system **108** may include one or more indexers **206**, depending on system access resources and requirements. As described further below, the indexers **206** retrieve data from their respective local data stores **208** as specified in the search request. The indexers and their respective data stores can comprise one or more storage devices and typically reside on the same system, though they may be connected via a local network connection.

[0172] If the request parameters of the received search request reference an external data collection, which is not accessible to the indexers **206** or under the management of the data intake and query system, then the search head **210** can access the external data collection through an External Result Provider (ERP) process **810**. An external data collection may be referred to as a "virtual index" (plural, "virtual indices"). An ERP process provides an interface through which the search head **210** may access virtual indices.

[0173] Thus, a search reference to an index of the system relates to a locally stored and managed data collection. In contrast, a search reference to a virtual index relates to an externally stored and managed data collection, which the search head may access through one or more ERP processes **810**, **812**. FIG. **8** shows two ERP processes **810**, **812** that connect to respective remote (external) virtual indices, which are indicated as a Hadoop or another system **814** (e.g., Amazon S3, Amazon EMR, other Hadoop Compatible File Systems (HCFS), etc.) and a relational database management system (RDBMS) **816**. Other virtual indices may include other file organizations and protocols, such as Structured Query Language (SQL) and the like. The ellipses between the ERP processes **810**, **812** indicate optional additional ERP processes of the data intake and query system **108**. An ERP process may be a computer process that is initiated or spawned by the search head **210** and is executed by the search data intake and query system **108**. Alternatively or additionally, an ERP process may be a process spawned by the search head **210** on the same or different host system as the search head **210** resides.

[0174] The search head **210** may spawn a single ERP process in response to multiple virtual indices referenced in a search request, or the search head may spawn different ERP processes for different virtual indices. Generally, virtual indices that share common data configurations or protocols may share ERP processes. For example, all search query references to a Hadoop file system may be processed by the same ERP process, if the ERP process is suitably configured. Likewise, all search query references to an SQL database may be processed by the same ERP process. In addition, the search head may provide a common ERP process for common external data source types (e.g., a common vendor may utilize a common ERP process, even if the vendor includes different data storage system types, such as Hadoop and SQL). Common indexing schemes also may be handled by common ERP processes, such as flat text files or Weblog files.

[0175] The search head **210** determines the number of ERP processes to be initiated via the use of configuration parameters that are included in a search request message. Generally, there is a one-to-many relationship between an external results provider "family" and ERP processes. There is also a one-to-many relationship between an ERP process and corresponding virtual indices that are referred to in a search request. For example, using RDBMS, assume two independent instances of such a system by one vendor, such as one RDBMS for production and another RDBMS used for development. In such a situation, it is likely preferable (but optional) to use two ERP processes to maintain the independent operation as between production and development data. Both of the ERPs, however, will belong to the same family, because the two RDBMS system types are from the same vendor.

[0176] The ERP processes **810**, **812** receive a search request from the search head **210**. The search head may optimize the received search request for execution at the respective external virtual index. Alternatively, the ERP process may receive a search request as a result of analysis performed by the search head or by a different system process. The ERP processes **810**, **812** can communicate with the search head **210** via conventional input/output routines (e.g., standard in/standard out, etc.). In this way, the ERP process receives the search request from a client device such that the search request may be efficiently executed at the corresponding external virtual index.

[0177] The ERP processes **810**, **812** may be implemented as a process of the data intake and query system. Each ERP process may be provided by the data intake and query system, or may be provided by process or application providers who are independent of the data intake and query system. Each respective ERP process may include an interface application installed at a computer of the external result provider that ensures proper communication between the search support system and the external result provider. The ERP processes **810**, **812** generate appropriate search requests in the protocol and syntax of the respective virtual indices **814**, **816**, each of which corresponds to the search request received by the search head **210**. Upon receiving search results from their corresponding virtual indices, the respective ERP process passes the result to the search head **210**, which may return or display the results or a processed set of results based on the returned results to the respective client device.

[0178] Client devices **804** may communicate with the data intake and query system **108** through a network interface **820**, e.g., one or more LANs, WANs, cellular networks, intranetworks, and/or internetworks using any of wired, wireless, terrestrial microwave, satellite links, etc., and may include the public Internet.

[0179] The analytics platform utilizing the External Result Provider process described in more detail in U.S. Pat. No. 8,738,629, entitled "External Result Provided Process For RetriEving Data Stored Using A Different Configuration Or Protocol", issued on 27 May 2014, U.S. Pat. No. 8,738,587, entitled "PROCESSING A SYSTEM SEARCH REQUEST BY RETRIEVING RESULTS FROM BOTH A NATIVE INDEX AND A VIRTUAL INDEX", issued on 25 Jul. 2013, U.S. patent application Ser. No. 14/266,832, entitled "PROCESSING A SYSTEM SEARCH REQUEST ACROSS DISPARATE DATA COLLECTION SYSTEMS", filed on 1 May 2014, and U.S. patent application Ser. No. 14/449,144, entitled "PROCESSING A SYSTEM SEARCH REQUEST INCLUDING EXTERNAL DATA SOURCES", filed on 31 Jul. 2014, each of which is hereby incorporated by reference in its entirety for all purposes.

[0180] 2.13.1. ERP Process Features

[0181] The ERP processes described above may include two operation modes: a streaming mode and a reporting mode. The ERP processes can operate in streaming mode only, in reporting mode only, or in both modes simultaneously. Operating in both modes simultaneously is referred to as mixed mode operation. In a mixed mode operation, the ERP at some point can stop providing the search head with streaming results and only provide reporting results thereafter, or the search head at some point may start ignoring streaming results it has been using and only use reporting results thereafter.

[0182] The streaming mode returns search results in real time, with minimal processing, in response to the search request. The reporting mode provides results of a search request with processing of the search results prior to providing them to the requesting search head, which in turn provides results to the requesting client device. ERP operation with such multiple modes provides greater performance flexibility with regard to report time, search latency, and resource utilization.

[0183] In a mixed mode operation, both streaming mode and reporting mode are operating simultaneously. The streaming mode results (e.g., the raw data obtained from the external data source) are provided to the search head, which can then process the results data (e.g., break the raw data into events, timestamp it, filter it, etc.) and integrate the results data with the results data from other external data sources, and/or from data stores of the search head. The search head performs such processing and can immediately start returning interim (streaming mode) results to the user at the requesting client device; simultaneously, the search head is waiting for the ERP process to process the data it is retrieving from the external data source as a result of the concurrently executing reporting mode.

[0184] In some instances, the ERP process initially operates in a mixed mode, such that the streaming mode operates to enable the ERP quickly to return interim results (e.g., some of the raw or unprocessed data necessary to respond to a search request) to the search head, enabling the search head to process the interim results and begin providing to the client or search requester interim results that are responsive to the query. Meanwhile, in this mixed mode, the ERP also operates concurrently in reporting mode, processing portions of raw data in a manner responsive to the search query. Upon determining that it has results from the reporting mode available to return to the search head, the ERP may halt processing in the mixed mode at that time (or some later time) by stopping the return of data in streaming mode to the search head and switching to reporting mode only. The ERP at this point starts sending interim results in reporting mode to the search head, which in turn may then present this processed data responsive to the search request to the client or search requester. Typically the search head switches from using results from the ERP's streaming mode of operation to results from the ERP's reporting mode of operation when the higher bandwidth results from the reporting mode outstrip the amount of data processed by the search head in the streaming mode of ERP operation.

[0185] A reporting mode may have a higher bandwidth because the ERP does not have to spend time transferring data to the search head for processing all the raw data. In addition, the ERP may optionally direct another processor to do the processing.

[0186] The streaming mode of operation does not need to be stopped to gain the higher bandwidth benefits of a reporting mode; the search head could simply stop using the streaming mode results—and start using the reporting mode results—when the bandwidth of the reporting mode has caught up with or exceeded the amount of bandwidth provided by the streaming mode. Thus, a variety of triggers and ways to accomplish a search head's switch from using streaming mode results to using reporting mode results may be appreciated by one skilled in the art.

[0187] The reporting mode can involve the ERP process (or an external system) performing event breaking, time stamping, filtering of events to match the search query request, and calculating statistics on the results. The user can request particular types of data, such as if the search query itself involves types of events, or the search request may ask for statistics on data, such as on events that meet the search request. In either case, the search head understands the query language used in the received query request, which may be a proprietary language. One example query language is Splunk Processing Language (SPL) developed by the assignee of the application, Splunk Inc. The search head typically understands how to use that language to obtain data from the indexers, which store data in a format used by the SPLUNK® Enterprise system.

[0188] The ERP processes support the search head, as the search head is not ordinarily configured to understand the format in which data is stored in external data sources such as Hadoop or SQL data systems. Rather, the ERP process performs that translation from the query submitted in the search support system's native format (e.g., SPL if SPLUNK® ENTERPRISE is used as the search support system) to a search query request format that will be accepted by the corresponding external data system. The external data system typically stores data in a different format from that of the search support system's native index format, and it utilizes a different query language (e.g., SQL or MapReduce, rather than SPL or the like).

[0189] As noted, the ERP process can operate in the streaming mode alone. After the ERP process has performed the translation of the query request and received raw results from the streaming mode, the search head can integrate the returned data with any data obtained from local data sources (e.g., native to the search support system), other external data sources, and other ERP processes (if such operations were required to satisfy the terms of the search query). An advantage of mixed mode operation is that, in addition to streaming mode, the ERP process is also executing concur-

rently in reporting mode. Thus, the ERP process (rather than the search head) is processing query results (e.g., performing event breaking, timestamping, filtering, possibly calculating statistics if required to be responsive to the search query request, etc.). It should be apparent to those skilled in the art that additional time is needed for the ERP process to perform the processing in such a configuration. Therefore, the streaming mode will allow the search head to start returning interim results to the user at the client device before the ERP process can complete sufficient processing to start returning any search results. The switchover between streaming and reporting mode happens when the ERP process determines that the switchover is appropriate, such as when the ERP process determines it can begin returning meaningful results from its reporting mode.

[0190] The operation described above illustrates the source of operational latency: streaming mode has low latency (immediate results) and usually has relatively low bandwidth (fewer results can be returned per unit of time). In contrast, the concurrently running reporting mode has relatively high latency (it has to perform a lot more processing before returning any results) and usually has relatively high bandwidth (more results can be processed per unit of time). For example, when the ERP process does begin returning report results, it returns more processed results than in the streaming mode, because, e.g., statistics only need to be calculated to be responsive to the search request. That is, the ERP process doesn't have to take time to first return raw data to the search head. As noted, the ERP process could be configured to operate in streaming mode alone and return just the raw data for the search head to process in a way that is responsive to the search request. Alternatively, the ERP process can be configured to operate in the reporting mode only. Also, the ERP process can be configured to operate in streaming mode and reporting mode concurrently, as described, with the ERP process stopping the transmission of streaming results to the search head when the concurrently running reporting mode has caught up and started providing results. The reporting mode does not require the processing of all raw data that is responsive to the search query request before the ERP process starts returning results; rather, the reporting mode usually performs processing of chunks of events and returns the processing results to the search head for each chunk.

[0191] For example, an ERP process can be configured to merely return the contents of a search result file verbatim, with little or no processing of results. That way, the search head performs all processing (such as parsing byte streams into events, filtering, etc.). The ERP process can be configured to perform additional intelligence, such as analyzing the search request and handling all the computation that a native search indexer process would otherwise perform. In this way, the configured ERP process provides greater flexibility in features while operating according to desired preferences, such as response latency and resource requirements.

[0192] 2.14. It Service Monitoring

[0193] As previously mentioned, the SPLUNK® ENTERPRISE platform provides various schemas, dashboards and visualizations that make it easy for developers to create applications to provide additional capabilities. One such application is SPLUNK® IT SERVICE INTELLIGENCE™, which performs monitoring and alerting operations. It also includes analytics to help an analyst diagnose the root cause of performance problems based on large volumes of data stored by the SPLUNK® ENTERPRISE system as correlated to the various services an IT organization provides (a service-centric view). This differs significantly from conventional IT monitoring systems that lack the infrastructure to effectively store and analyze large volumes of service-related event data. Traditional service monitoring systems typically use fixed schemas to extract data from pre-defined fields at data ingestion time, wherein the extracted data is typically stored in a relational database. This data extraction process and associated reduction in data content that occurs at data ingestion time inevitably hampers future investigations, when all of the original data may be needed to determine the root cause of or contributing factors to a service issue.

[0194] In contrast, a SPLUNK® IT SERVICE INTELLIGENCE™ system stores large volumes of minimally-processed service-related data at ingestion time for later retrieval and analysis at search time, to perform regular monitoring, or to investigate a service issue. To facilitate this data retrieval process, SPLUNK® IT SERVICE INTELLIGENCE™ enables a user to define an IT operations infrastructure from the perspective of the services it provides. In this service-centric approach, a service such as corporate e-mail may be defined in terms of the entities employed to provide the service, such as host machines and network devices. Each entity is defined to include information for identifying all of the event data that pertains to the entity, whether produced by the entity itself or by another machine, and considering the many various ways the entity may be identified in raw machine data (such as by a URL, an IP address, or machine name). The service and entity definitions can organize event data around a service so that all of the event data pertaining to that service can be easily identified. This capability provides a foundation for the implementation of Key Performance Indicators.

[0195] One or more Key Performance Indicators (KPI's) are defined for a service within the SPLUNK® IT SERVICE INTELLIGENCE™ application. Each KPI measures an aspect of service performance at a point in time or over a period of time (aspect KPI's). Each KPI is defined by a search query that derives a KPI value from the machine data of events associated with the entities that provide the service. Information in the entity definitions may be used to identify the appropriate events at the time a KPI is defined or whenever a KPI value is being determined. The KPI values derived over time may be stored to build a valuable repository of current and historical performance information for the service, and the repository, itself, may be subject to search query processing. Aggregate KPIs may be defined to provide a measure of service performance calculated from a set of service aspect KPI values; this aggregate may even be taken across defined timeframes and/or across multiple services. A particular service may have an aggregate KPI derived from substantially all of the aspect KPI's of the service to indicate an overall health score for the service.

[0196] SPLUNK® IT SERVICE INTELLIGENCE™ facilitates the production of meaningful aggregate KPI's through a system of KPI thresholds and state values. Different KPI definitions may produce values in different ranges, and so the same value may mean something very different from one KPI definition to another. To address this, SPLUNK® IT SERVICE INTELLIGENCE™ implements a translation of individual KPI values to a common domain of

"state" values. For example, a KPI range of values may be 1-100, or 50-275, while values in the state domain may be 'critical,' warning,' 'normal,' and 'informational'. Thresholds associated with a particular KPI definition determine ranges of values for that KPI that correspond to the various state values. In one case, KPI values 95-100 may be set to correspond to 'critical' in the state domain. KPI values from disparate KPI's can be processed uniformly once they are translated into the common state values using the thresholds. For example, "normal 80% of the time" can be applied across various KPI's. To provide meaningful aggregate KPI's, a weighting value can be assigned to each KPI so that its influence on the calculated aggregate KPI value is increased or decreased relative to the other KPI's.

[0197] One service in an IT environment often impacts, or is impacted by, another service. SPLUNK® IT SERVICE INTELLIGENCE™ can reflect these dependencies. For example, a dependency relationship between a corporate e-mail service and a centralized authentication service can be reflected by recording an association between their respective service definitions. The recorded associations establish a service dependency topology that informs the data or selection options presented in a GUI, for example. (The service dependency topology is like a "map" showing how services are connected based on their dependencies.) The service topology may itself be depicted in a GUI and may be interactive to allow navigation among related services.

[0198] Entity definitions in SPLUNK® IT SERVICE INTELLIGENCE™ can include informational fields that can serve as metadata, implied data fields, or attributed data fields for the events identified by other aspects of the entity definition. Entity definitions in SPLUNK® IT SERVICE INTELLIGENCE™ can also be created and updated by an import of tabular data (as represented in a CSV, another delimited file, or a search query result set). The import may be GUI-mediated or processed using import parameters from a GUI-based import definition process. Entity definitions in SPLUNK® IT SERVICE INTELLIGENCE™ can also be associated with a service by means of a service definition rule. Processing the rule results in the matching entity definitions being associated with the service definition. The rule can be processed at creation time, and thereafter on a scheduled or on-demand basis. This allows dynamic, rule-based updates to the service definition.

[0199] During operation, SPLUNK® IT SERVICE INTELLIGENCE™ can recognize so-called "notable events" that may indicate a service performance problem or other situation of interest. These notable events can be recognized by a "correlation search" specifying trigger criteria for a notable event: every time KPI values satisfy the criteria, the application indicates a notable event. A severity level for the notable event may also be specified. Furthermore, when trigger criteria are satisfied, the correlation search may additionally or alternatively cause a service ticket to be created in an IT service management (ITSM) system, such as a systems available from ServiceNow, Inc., of Santa Clara, Calif.

[0200] SPLUNK® IT SERVICE INTELLIGENCE™ provides various visualizations built on its service-centric organization of event data and the KPI values generated and collected. Visualizations can be particularly useful for monitoring or investigating service performance. SPLUNK® IT SERVICE INTELLIGENCE™ provides a service monitoring interface suitable as the home page for ongoing IT service monitoring. The interface is appropriate for settings such as desktop use or for a wall-mounted display in a network operations center (NOC). The interface may prominently display a services health section with tiles for the aggregate KPI's indicating overall health for defined services and a general KPI section with tiles for KPI's related to individual service aspects. These tiles may display KPI information in a variety of ways, such as by being colored and ordered according to factors like the KPI state value. They also can be interactive and navigate to visualizations of more detailed KPI information.

[0201] SPLUNK® IT SERVICE INTELLIGENCE™ provides a service-monitoring dashboard visualization based on a user-defined template. The template can include user-selectable widgets of varying types and styles to display KPI information. The content and the appearance of widgets can respond dynamically to changing KPI information. The KPI widgets can appear in conjunction with a background image, user drawing objects, or other visual elements, that depict the IT operations environment, for example. The KPI widgets or other GUI elements can be interactive so as to provide navigation to visualizations of more detailed KPI information.

[0202] SPLUNK® IT SERVICE INTELLIGENCE™ provides a visualization showing detailed time-series information for multiple KPI's in parallel graph lanes. The length of each lane can correspond to a uniform time range, while the width of each lane may be automatically adjusted to fit the displayed KPI data. Data within each lane may be displayed in a user selectable style, such as a line, area, or bar chart. During operation a user may select a position in the time range of the graph lanes to activate lane inspection at that point in time. Lane inspection may display an indicator for the selected time across the graph lanes and display the KPI value associated with that point in time for each of the graph lanes. The visualization may also provide navigation to an interface for defining a correlation search, using information from the visualization to pre-populate the definition.

[0203] SPLUNK® IT SERVICE INTELLIGENCE™ provides a visualization for incident review showing detailed information for notable events. The incident review visualization may also show summary information for the notable events over a time frame, such as an indication of the number of notable events at each of a number of severity levels. The severity level display may be presented as a rainbow chart with the warmest color associated with the highest severity classification. The incident review visualization may also show summary information for the notable events over a time frame, such as the number of notable events occurring within segments of the time frame. The incident review visualization may display a list of notable events within the time frame ordered by any number of factors, such as time or severity. The selection of a particular notable event from the list may display detailed information about that notable event, including an identification of the correlation search that generated the notable event.

[0204] SPLUNK® IT SERVICE INTELLIGENCE™ provides pre-specified schemas for extracting relevant values from the different types of service-related event data. It also enables a user to define such schemas.

[0205] 3.0. Behavioral Based Device Clustering

[0206] As discussed above, behavioral based device clustering is the clustering of devices in a network according the

behavior of the devices. FIG. **9** illustrates a block diagram of an example data intake and query system in which an embodiment may be implemented. In FIG. **9**, the data sources **202**, data intake and query system **108**, forwarder **204**, indexer **206**, and search head **210** may be the same or similar to the like named components shown in FIG. **2**. The event data store **208** may be the same or similar to the data store **208** shown in FIG. **2**. Further, although fewer instances of the components are shown, any number of instances may exist. For example, the data sources may be in the hundreds or thousands. Similarly, the number of forwarders, indexers, and data stores may similarly scale.

[0207] As shown in FIG. **9**, the data intake and query system further include a device group data store **902** and an analyzer **904**. The data intake and query system **108** may correspond to a server group having multiple servers. Each server may, for example, correspond to hardware. Thus, the forwarder **204**, indexer **206**, data store **208**, search head **210**, device group data store **902**, and analyzer **904** may each be a server.

[0208] The device group data store **902** is any type of data store that includes functionality to store information about devices and device groups. For example, the device group data store **902** may be a database, one or more storage devices, memory, file system, or other storage. A device is any virtual or physical device on the network. For example, the device may be a virtual machine, a virtual network function, a terminal server, storage servers, gateways, routers, network bridges, modems, wireless access points, switches, hubs, repeaters, a laptop computer, a desktop computer, mobile device, or any other device connected to and managed and/or monitored by the network administrative devices. Devices in the device group data store **902** may be uniquely identified by one or more device identifiers in the device group data store **902**. For example, a device may be identified by a media access control (MAC) address, internet protocol (IP) address, assigned name, or other identifier.

[0209] The device group data store **902** further includes functionality to store device groups. A device group is a collection of one or more devices that are grouped together. Each device group is associated with a unique device group identifier. The device group data store **902** includes, for each device group, one or more device group identifiers that are unique to the device group. For example, the device group identifier may be any string of characters that are unique to the device group. In one or more embodiments, each device assigned to the device group is related to the device group identifier in the device group data store **902**.

[0210] Continuing with FIG. **9**, an analyzer **904** is communicatively connected to the device group data store **902**. The analyzer **904** may be hardware, software, or firmware, or any combination thereof that includes functionality to process event data to generate conclusions (e.g., anomalies, threat indicators, threats, or any combination thereof). The analyzer **904** may operate in real-time. "Real-time" computing, or "reactive computing", describes computer systems subject to a processing responsiveness restriction (e.g., in a service level objective (SLO) in a service level agreement (SLA)). In real-time processing, conclusions are reached substantially immediately following the receipt of input data such that the conclusions can be used to respond to the observed environment. The analyzer **904** continuously receives new incoming raw event data from the indexer **206** and reacts to each new incoming event by processing the event through the anomaly detector and the threat detector. Because of real-time processing, the analyzer **904** can begin to process a time slice of the unbounded stream prior to when a subsequent time slice from the unbounded stream becomes available. The analyzer **904** may further analyze historical data. In such a scenario, the analyzer **904** may use event data obtained from queries submitted to the search head **210**.

[0211] The analyzer **904** includes a classifier **906**, anomaly detector **908**, and a threat detector **910**. The classifier **906** includes functionality to process events in a network traffic log and classify devices into device groups based on the events. In one or more embodiments, the classifier **906** is configured to apply a topic modeling algorithm to group devices. A topic modeling algorithm relates topics to objects. Specifically, for each object, the topic modeling algorithm extracts features of the object, determines a set of topics for a set of features, and generates a set of scores for devices and topics. An example of a topic modeling algorithms is Latent Dirichlet Analysis (LDA). In one or more embodiments, the objects in the topic modeling algorithm are devices, and the topics are device groups. In some embodiments in which multiple stage classification is used, such as described below in reference to FIG. **10**, the topics output in the first stage of applying the topic modeling algorithm are combined with the topics output in the second stage to form the device group. More details of the multiple stage embodiments are described below with reference to FIGS. **10**, **12**, and **15**.

[0212] An anomaly detector **908** is hardware, software, firmware, or any combination thereof that includes functionality to detect anomalies. In this description, an "anomaly" is a detected variation from an expected pattern of behavior on the part of an entity. In security, an anomaly may or may not be indicative of a threat. An anomaly represents an action of possible concern, which may be actionable or warrant further investigation. An anomaly is an observable or detectable fact, or data representing such fact.

[0213] Continuing with the anomaly detector **908**, the anomaly detector **908** includes functionality to process events in real time and/or based on historical events. In one or more embodiments, the anomaly detector **908** operates based on the behavior groups of the corresponding device groups. Rather than tracking and maintaining information about the behavior of each device individually for anomaly detection, the behavior of the device group may be maintained and tracked. Specifically, because devices are grouped based on the devices' behaviors, deviations from the behaviors for a device may be identified based on deviations of the behavior of the device group to which the device is assigned. With less data being stored and processed, the anomaly detector **908** is faster at detecting anomalies and the device group data store **902** may have less memory utilization.

[0214] A threat detector **910** includes functionality to detect threats based on detected anomalies. A "threat" is an interpretation of one or more anomalies and/or threat indicators. Threat indicators and threats are escalations of actions of concern. For example, a threat may be an intrusion, a failure in a device or a component, unbalanced resource utilization of devices, or other threat to the operations of the network. The threat detector **910** includes functionality to detect threats in real time and/or based on

historical events. The threat detector **910** may detect threats based on the output of the anomaly detector **908**.

[0215] As an example of scale, hundreds of millions of packets of incoming data from various data sources may be analyzed to yield **100** anomalies, which may be further analyzed to yield **10** threat indicators, which may again be further analyzed to yield one or two threats. This manner of data scaling is one of the reasons the security platform can provide anomaly and threat detection in a real-time manner.

[0216] Although not shown in FIG. **9**, the analyzer **904** may further include a device manager. In other embodiments, the device manager may be, in whole or in part, separate from the analyzer **904** and/or the data intake and query system. The device manager includes functionality to manage devices based on the device groups created by the classifier **906**. For example, the device manager is configured to initiate or turn on devices and/or services on the devices, deploy tools, applications, and services to the devices, adjust configurations of the devices, and perform other management tasks. The device manager performs most of the management tasks on a per device group basis. In other words, management operations performed by the device manager are selected by the device manager to be performed for the entire device group. By grouping devices based on behavior, one or more embodiments simplify the management operations of the device.

[0217] FIG. **10** shows a diagram of a classifier **906**, event data store **208**, and device group data store **902**. The classifier **906**, event data store **208**, and device group data store **902** may correspond to like named components of FIG. **9**. The various components of FIG. **10** may be part of the data intake and query system as described above or a distinct system. As shown in FIG. **10**, the event data store **208** includes functionality to store events **1002**. As discussed above, an event includes a portion of machine-generated data and is associated with a specific point in time. Events in the event data store **208** is a recordation of an action performed by one or more devices. The events include a network traffic log having network traffic log entries **1004**. The network traffic log is a recording of traffic events in the network, such as the communication of packets in the network. Examples of network traffic logs include a firewall log recording network traffic events through one or more firewalls, a router log that records packets routed by one or more routers, or other log of network traffic. The network traffic log includes network traffic log entries **1004**. Each entry is a distinct event. For example, each entry may correspond to a single packet processed by the firewall on the network.

[0218] Although not shown, additional data sources may be used by the classifier. For example, the additional data may include endpoint logs, operating system logs, and various types of application logs. The endpoint logs may provide information about processes running in a machine, and the operating system of a device. The operating system logs may provide information about the activity in the device. Application logs, such as database logs may be used to provide information about the applications that are executing on each device.

[0219] As further shown in FIG. **10**, the device group data store **902** includes functionality to store aggregated feature matrix **1006** and device group information **1008**. A feature is a piece of information extracted from the events. For example, a feature may be whether a device in the event is the destination or the source of the packet, the port number, protocol used, application name, application type and other information that may be extracted from an event. An aggregated feature matrix **1006** is an aggregation of features on a per device basis. Each entry in the aggregated feature matrix **1006** is for a feature and a device pair and has a value of the total number of events that have the feature for the device. For example, a row of the aggregated feature matrix may be for a single device. A column of the aggregated feature matrix is for a single feature. Thus, by way of a more specific example, the value of entry (device DV1, feature F4) is the total number of events in which device DV1 had feature F4. In the topic modeling algorithm, each event referencing a device is a sentence, the aggregated feature set is the term set, and all the sentences related with the device pooled together is a document.

[0220] An aggregated feature matrix may have defined subsets (i.e., sub-matrices) that include application behavior subsets and traffic behavior subsets. An application behavior subset is the subset of features of the aggregated feature matrix that relate to the usage of applications as reflected in the events. For example, the application behavior subsets may include the various application types of the applications reflected in the events. The traffic behavior subset is the subset of features in the aggregated feature matrix that define the path of the communications through the network. For example, the traffic behavior subset may include the size of the packet, the port number, whether the device is the source or the designation, and other information.

[0221] Although the term matrix is used in the application, the term matrix is for conceptual purposes only to refer to a two or more dimensional structure. Any storage structure for storing such a conceptual structure may be used without departing from the scope of the disclosure.

[0222] Continuing with the device group data store **902**, the device group information **1008** is information describing the device groups. For example, the device group information for a device group includes the device group identifier of the device group, device identifiers of devices in the device group, and behavior group identifiers of the behavior groups matching the device group. Device groups and behavior groups are discussed in further detail in FIG. **11**.

[0223] Continuing with FIG. **10**, the classifier **906** is communicatively connected to the event data store **208** and the device group data store **902**. The classifier **906** includes a traffic behavior stage **1010** and an application behavior stage **1012**. The traffic behavior stage **1010** includes functionality to group devices using the traffic behavior subsets of the aggregated feature matrix. The application behavior stage **1012** includes functionality to group devices based on the application behavior subsets of the aggregated feature matrix.

[0224] FIG. **11** illustrates a block diagram of device groups (e.g., device group X (**1102**), device group Y (**1104**)) in the device group data store in accordance with disclosed embodiments. As shown in FIG. **11**, a device group (e.g., device group X (**1102**), device group Y (**1104**)) includes one or more device group identifiers (e.g., device group X identifier (**1106**), device group Y identifier (**1108**)) and zero or more member device identifiers (e.g., member device identifiers X (**1110**), member device identifiers Y (**1112**)). The device group identifier is the same as discussed above with respect to FIGS. **9** and **10**, and uniquely identifies the device group. A device group has zero or more member

devices. A member device is a device that is classified by the classifier as being in the respective device group. The member device identifiers (e.g., member device identifiers X (**1110**), member device identifiers Y (**1112**)) are the same as the device identifiers discussed above with reference to FIGS. **9** and **10** for the member devices of the device group.

[0225] A device group is defined based on one or more behavior groups. A behavior group is an aspect of the behavior of one or more devices. In other words, a behavior group is a part or characteristic of a behavior of one or more devices. By of some examples, a behavior group may be an application type of applications used by the devices, whether the devices behaves like a domain controller or other service, the amount of internal traffic volume produced or consumed by the devices, and other aspects of the behavior of devices that is reflected in the network traffic log. In one or more embodiments, a behavior group corresponds to a feature extracted from the events.

[0226] In some embodiments, a device group is in a one to one mapping with behavior groups. In such a scenario, each device group has a single corresponding behavior group. In some embodiments, a device group may have one or more corresponding behavior groups. Specifically, each device group may match a single behavior group or multiple behavior groups. Either embodiment may be implemented with the remainder of the description without departing from the scope of the claims.

[0227] As shown in FIG. **11**, a device group may have one or more corresponding behavior group scores defined for one or more behavior groups. For example, device group X **1102** has behavior group A score **1114**, behavior group B score **1116**, and behavior group K score **1118** while device group Y **1104** has behavior group A score **1120**, behavior group B score **1122**, and behavior group N score **1124**. The combination of behavior group scores assigned to a device group is unique amongst the device groups in accordance with one or more embodiments. A behavior group score is a value defining the degree in which member devices in the device group have behaviors matching the behavior group. In one or more embodiments, a score is a scaled value on a range of possible values. For example, a score may be a probability value assigned by the classifier. Different types of scoring methodologies may be used for the behavior group score. Below are a few examples of scoring methodologies. The various examples describe below may be combined in virtually any manner.

[0228] In an example, a behavior group score assigned to a device group is a single value. The single value may be a zero or non-zero score. A score of zero is indicative that the member devices in the device group do not exhibit any features of the behavior group, while a non-zero score is indicative that the member devices exhibit some of the features.

[0229] In another example, a behavior group score assigned to a device group is a percentage value. The percentage value is the amount that the device group exhibits the behavior of the behavior group. In the example, the device group may be a mixture of behavior groups, whereby the total of the mixture is one. By way of a more specific example, a set of behavior group scores that are 0.6 for behavior group 1, 0.3 for behavior group 2, 0.1 for behavior group 3 indicates that device group exhibits 0.6 of the attributes of behavior group 1, 0.3 of the attributes of behavior group 2, 0.1 of the attributes of behavior group 3.

[0230] In another example, the behavior group score may be of nominal levels that is a value in a discrete set of values (e.g., "zero" for not belong to the device group, "low" for low degree of belonging to the device group, "medium" for medium degree of belonging to the device group, and "high" for high degree of belonging to the device group). Although the above is a few example methodologies of behavior group scoring, other methodologies may be used without departing from the scope describe herein.

[0231] Turning to FIG. **12**, FIG. **12** shows a hierarchical arrangement of device groups. As shown in FIG. **12**, devices may be clustered into traffic behavior device groups (e.g., traffic behavior device group **1202**). The devices in the traffic behavior device group may be further clustered according to application behavior subsets into application behavior subgroups (e.g., application behavior subgroup **1204**, application behavior subgroup **1206**). Each application behavior subgroup is an independent device group within the traffic behavior device group. Specifically, the application behavior subgroup has the application behavior group scores of the application behavior subgroup as well as the traffic behavior group scores of the traffic behavior device group. Although FIG. **12** shows only two layers in a hierarchy, additional layers may exist without departing from the scope of the disclosure.

[0232] FIGS. **13-16** show flowcharts in accordance with disclosed embodiments. While the various steps in these flowcharts are presented and described sequentially, one of ordinary skill will appreciate that some or all of the steps may be executed in different orders, may be combined or omitted, and some or all of the steps may be executed in parallel. Furthermore, the steps may be performed actively or passively. For example, some steps may be performed using polling or be interrupt driven in accordance with one or more embodiments of the invention. By way of an example, determination steps may not require a processor to process an instruction unless an interrupt is received to signify that condition exists in accordance with one or more embodiments of the invention. As another example, determination steps may be performed by performing a test, such as checking a data value to test whether the value is consistent with the tested condition in accordance with one or more embodiments of the invention.

[0233] Turning to FIG. **13**, FIG. **13** is a flow diagram that illustrates behavior based device clustering in accordance with disclosed embodiments. In Block **1302**, a network traffic log of devices in a network is received. As network traffic, such as individual packets, is transmitted through a data source device of the network, the data source device processes the network traffic and generates events recording the processing of the network traffic. The data source device stores the events in a network traffic log. The data source sends the network traffic log to the data intake and query system in whole or in part. For example, as events are generated by the data source of the network traffic log, the events may be sent to the event data store. As another example, a collection of log entries or the entire network traffic log for a timespan may be sent to the event data store. The network traffic log may be transmitted using various standard network communication protocols via the network to the event data store. The network traffic log may be preprocessed to remove extraneous information. For example, the zone and timestamp features may be removed from the network traffic log to remove noise.

[0234] In Block **1304**, features of the devices are extracted from the network traffic log. In one or more embodiments, the features that are extracted are predefined features of the events of the network traffic log. In one or more embodiments, each event is parsed, and device identifiers, protocols, and other features are extracted from the events based on the predefined features. The events may record feature name and feature value pairs for each feature in the network traffic log. In such a scenario, the features may be extracted based on being the value in the pair matching the feature name of the predefined features. As another example, features may be extracted based on position within the event.

[0235] In Block **1306**, on a per device basis, the features are aggregated into aggregated feature matrix for the devices. For each device and for each feature, a total number of events having the device and feature of the device are determined. For example, if the feature is that the device is the destination of a packet, then the total number of events recording that the device is the destination of the packet is calculated.

[0236] Although not shown in FIG. **13**, Blocks **1304** and **1306** may be performed concurrently. For example, for each event in the network device log, the device identifiers of any devices referenced in the network traffic log may be extracted from the event. Features are extracted from the event and used to update totals in the aggregated feature matrix for the devices matching the device identifiers, such as by incrementing the totals in the aggregated feature matrix for each matching feature by one. Stated another way, the data store may be queried for a subset of the timestamp entries that each include the device identifier of a device. The features may be aggregated in the subset of the timestamp entries into a feature set for the device. The feature set is added to the aggregated feature matrix. For example, the feature set may be the row of values for the device.

[0237] By way of another example, Blocks **1304** and **1306** may be performed by sending a query to the search head, and having the query processed using the pipeline described above with reference to FIG. **6**. For example, the analyzer may send the query to the search head to extract features from the network traffic log on a per device basis and generate an aggregated feature set for the device. In such a scenario, generating the aggregated feature matrix may be performed as discussed above with reference to FIGS. **1-8**.

[0238] In Block **1308**, by applying the topic modeling algorithm, the devices are clustered into device groups according to one or more behavior groups. A topic modeling algorithm automatically learns the distribution of topics per device and the distribution of behavior groups per topic. Based on the output of the topic modeling algorithm, devices are grouped into device groups. Below is a discussion of applying the topic modeling algorithm and clustering devices into device groups in accordance with one or more embodiments.

[0239] As described above, the aggregated feature matrix, has, for each device, feature pair, the total number of events in which the device has the feature in the event. By applying the topic modeling algorithm, in a first phase, topics are identified, and feature scores are assigned to the topics. If the classifier is a single stage classifier, each topic is a device group. Thus, in the first phase, a topic feature matrix may be generated that has topics on one axis and features on another axis. The values in the topic feature matrix is the feature score for the topic. The topic modeling algorithm learns the topics based on the distribution of the features in the aggregated feature matrix. By way of an example, the topic modeling algorithm calculates, for each topic, using the aggregated feature set, a set of posterior probabilities that each behavior group is included in the topic. Further processing may be performed to limit the number of topics and/or reduce the size of the matrix. For example, the further processing may be to remove topics that do not have a feature satisfying a minimum score threshold and/or to remove features that do not satisfy a minimum score threshold. By way of another example, the further processing may be used to limit the number of topics to a maximum number.

[0240] In a second phase, the distribution of devices to the topics are learned by the topic modeling algorithm. Specifically, for each device, a set of posterior probabilities that the device is a member of each topic learned in the first phase is determined using the aggregated feature matrix. Thus, a device topic matrix may be generated that has devices on one axis and topics on the other axis. The values in the device topic matrix is the posterior probability that the device is in the topic. Stated another way the values of the device topic matrix may be referred to as the device topic score for a particular device and a particular topic. Thus, the values for device A of 0.3 topic X, 0.3 topic Y, and 0.4 topic Z indicates that device A has a posterior probability of 0.3 being in topic X, 0.3 of being in topic Y, and 0.4 of being in topic Z.

[0241] From the output of the topic modeling algorithm, the devices are clustered. For example, the topics may correspond to device groups. The features having non-zero feature scores or a greater than a minimum score in the topic feature matrix are each behavior groups, and the corresponding scores are the behavior group scores. A Bayesian classifier may be applied to determine the device group of each device. From the posterior probability determined in the device topic matrix, each device is classified into a topic by the Bayesian classifier. In a single stage classification, the topic to which the device is classified is the device group of the device. Multistage classification and relation to topics is described below in reference to FIG. **15**.

[0242] In Block **1310**, the devices on the network are assigned to device groups to obtain an assignment. The clustering determines the device group for each device. By iterating through the assigned devices, for each device group, the member devices may be identified, and corresponding member device identifiers related to the device group identifier. The device group information is stored and used to manage the devices. The assignment may be performed by sending the device group information to a network administrator or network administrator device. The network administrator or network administrator device may store in a configuration file of a network management server, the device group identifier and the member device identifiers of the device group. Accordingly, various operations to manage the network as described above with reference to FIG. **9** may be performed.

[0243] FIG. **14** is a flow diagram that illustrates how features may be extracted based on application type in accordance with the disclosed embodiments. In Block **1402**, an application type of an application identifier in a log entry associated with the device is identified. The network log entry includes a unique identifier of the application. Using the extraction rules, the peers may extract the application identifier from the log entry, such as based on feature name,

feature value pair or position. The application identifier may be compared with another distinct data source to identify the application type matching the application identifier. By way of some examples, the distinct data source may relate ad-selfservice", "bluecoat-auth-agent", "checkpoint-client-auth", and other such application identifiers to authservice (authentication service) application type; "carbonite", "crashplan", "ibackup", "ironmountain-connected", and other such application identifiers to the backup application type; "cassandra", "dabbledb", "db2", "expand hbase", "gds-db", and other such application identifiers to the database application type, and so forth. Thus, the application type matching the application identifier may be determined.

[0244] Continuing with FIG. **14**, in Block **1406**, a total number of instances of an application matching the application type in the network traffic log is calculated. For the application type and the device identifier, the current count is incremented in the aggregated feature set.

[0245] In Block **1408**, the number is stored in an aggregated feature set for the device. In other words, the aggregated feature set is updated for the application type and the device identifier. The flow of FIG. **14** is repeated for each identified application instance in the log.

[0246] In some embodiments, the flow of FIG. **14** may be performed by the analyzer sending the query to the search head. The search head may then process the query by, sending a search query to the peers to obtain, from log entries, having the device identifier, the application identifier. Then the search head may obtain the application type for each application identifier and aggregate the number of instance of each application type determined.

[0247] FIG. **15** is a flow diagram that illustrates how to create nested device groups in accordance with disclosed embodiments. In Block **1502**, the traffic behavior subset of the aggregated feature matrix that correspond to traffic behavior are selected. In one or more embodiments, the traffic behavior subset is the portion of the aggregated feature matrix corresponding to a predefined set of features in the aggregated feature matrix. Thus, the traffic behavior subsets may be directly access from the aggregated feature matrix based on the definition.

[0248] In Block **1504**, the topic modeling algorithm is applied to the traffic behavior subset to obtain traffic behavior device groups. The topic modeling algorithm is applied along with a Bayesian classifier to assign devices to traffic behavior device groups. A same or similar technique as described above with reference to Block **1308** may be used to assign devices to traffic behavior device groups. In particular, in the first phase and the second phase described with reference to Block **1308**, only the portion of the aggregated feature matrix corresponding to the traffic behavior subset is used. In the clustering phase described in reference to Block **1308**, the topics are the traffic behavior device groups and the features described in Block **1308** are the traffic behavior groups. Thus, the traffic behavior device groups may have one or more devices as determined by the Bayesian classifier while clustering the devices.

[0249] The next stage may be performed to determine the subgroups based on application behavior of each device. In Block **1506**, the application behavior subset of the aggregated feature matrix that correspond to application behavior are selected. Obtaining the application behavior subsets may be performed in a similar manner to Block **1502**. The application behavior subset obtained includes the number of each application type in the network traffic log for each device. In one or more embodiments, the application behavior subset is obtained on a per traffic behavior device group basis. In other words, multiple application behavior sub-matrices (i.e., subsets) are generated (e.g., one for each traffic behavior device group). Each of the application behavior subsets are used individually for processing in Block **1504**.

[0250] In Block **1504**, the topic modeling algorithm is applied, per traffic behavior device group, to the application behavior subset to obtain application behavior subgroups. In other words, for each traffic behavior device group, the topic modeling algorithm is applied individually. Thus, the topic modeling algorithm is performed multiple times. Accordingly, subgroups of the traffic behavior device groups are obtained. The topic modeling algorithm is applied along with a Bayesian classifier to assign devices in a particular traffic behavior device group to application behavior device groups. A same or similar technique as described above with reference to Block **1308** may be used to assign devices to application behavior device subgroups. In particular, only the portion of the aggregated feature matrix corresponding to the application behavior subset, for the devices in the traffic behavior device group, is used in the first phase and the second phase described with reference to Block **1308**. In the clustering phase described in reference to Block **1308**, the topics are the application behavior device subgroups groups and the features described in Block **1308** are the application behavior groups.

[0251] To determine the cluster for the devices, each application behavior subgroup is a device group having member devices as the members of the subgroup. The feature scores of the traffic behavior groups and the application behavior group may be concatenated to form the behavior group and behavior group scores of the device group.

[0252] Although FIG. **15** shows only two stages of classification, more than two stages may be performed based on additional partitioning of features or additional types of features.

[0253] The following examples are for explanatory purposes only and not intended to limit the scope of disclosed embodiments. FIG. **16** is an example diagram of a device group clustering in accordance with disclosed embodiments. As shown in FIG. **16**, Device Group 3 **1602** corresponds to multiple behavior groups (e.g., traffic volume service internal to internal, application of type domain controller for a destination, and application of type authentication service for a destination), each having corresponding behavior group scores (e.g., 0.35, 0.56, and 0.42, respectively) as shown in Box **1604**. The member devices of device group 3 may be identified by the devices' corresponding internet protocol address or device name as shown in Box **1606**. The member devices of device group 3 in Box **1606** each match the behavior groups and behavior group scores of Box **1604**.

[0254] Continuing with FIG. **16**, the system may have identified additional behavior groups based on the network traffic log, such as device group 8 shown in Box **1610**. As shown in Box **1612**, like Device Group 3, Device Group 8 also has behavior group traffic volume service internal to internal. Additionally, Device Group 8 has a behavior group of application of type database for the destination as shown in Box **1612**. Device Group 8 has different member devices as shown in Box **1614**.

[0255] FIG. **17** illustrates an example Graphical User Interface (GUI) **1700** in accordance with disclosed embodiments. For a device group, the GUI **1700** displays the device group identifier **1702**, the member device identifiers **1704**, and a chart **1704** for the behavior groups in the device group. In the example shown, the behavior group scores are on a scale from 0 to 4 indicating a contribution level of the behavior group to the device group. Additionally, a section **1708** shows details of the behavior group scores including for behavior groups having zero score.

[0256] 4.0 Hardware

[0257] The various components of the figures may be a computing system or implemented on a computing system. For example, the operations of the data stores, indexers, search heads, host device(s), client devices, data intake and query systems, data sources, external resources, and/or any other component shown and/or described above may be performed by a computing system. A computing system may include any combination of mobile, desktop, server, router, switch, embedded device, or other types of hardware. For example, the computing system may include one or more computer processors, non-persistent storage (e.g., volatile memory, such as random access memory (RAM), cache memory), persistent storage (e.g., a hard disk, an optical drive such as a compact disk (CD) drive or digital versatile disk (DVD) drive, a flash memory, etc.), a communication interface (e.g., Bluetooth interface, infrared interface, network interface, optical interface, etc.), and numerous other elements and functionalities. The computer processor(s) may be an integrated circuit for processing instructions. For example, the computer processor(s) may be one or more cores or micro-cores of a processor. The computing system may also include one or more input devices, such as a touchscreen, keyboard, mouse, microphone, touchpad, electronic pen, or any other type of input device.

[0258] The computing system may be connected to or be a part of a network. For example, the network may include multiple nodes. Each node may correspond to a computing system, such as the computing system, or a group of nodes combined may correspond to the computing system. By way of an example, embodiments of the disclosure may be implemented on a node of a distributed system that is connected to other nodes. By way of another example, embodiments of the disclosure may be implemented on a distributed computing system having multiple nodes, where each portion of the disclosure may be located on a different node within the distributed computing system. Further, one or more elements of the aforementioned computing system may be located at a remote location and connected to the other elements over a network.

[0259] The node may correspond to a blade in a server chassis that is connected to other nodes via a backplane. By way of another example, the node may correspond to a server in a data center. By way of another example, the node may correspond to a computer processor or micro-core of a computer processor with shared memory and/or resources.

[0260] The nodes in the network may be configured to provide services for a client device. For example, the nodes may be part of a cloud computing system. The nodes may include functionality to receive requests from the client device and transmit responses to the client device. The client device may be a computing system. Further, the client device may include and/or perform all or a portion of one or more embodiments of the disclosure.

[0261] Software instructions in the form of computer readable program code to perform embodiments of the disclosure may be stored, in whole or in part, temporarily or permanently, on a non-transitory computer readable medium such as a CD, DVD, storage device, a diskette, a tape, flash memory, physical memory, or any other computer readable storage medium. Specifically, the software instructions may correspond to computer readable program code that, when executed by a processor(s), is configured to perform one or more embodiments of the disclosure.

[0262] While the above figures show various configurations of components, other configurations may be used without departing from the scope of the disclosure. For example, various components may be combined to create a single component. As another example, the functionality performed by a single component may be performed by two or more components.

[0263] While the invention has been described with respect to a limited number of embodiments, those skilled in the art, having benefit of this disclosure, will appreciate that other embodiments can be devised which do not depart from the scope of the invention as disclosed herein. Accordingly, the scope of the invention should be limited only by the attached claims.

What is claimed is:

1. A method comprising:

receiving a network traffic log of a plurality of devices in a network;

extracting, from the network traffic log, a plurality of features of the plurality of devices;

aggregating, per device of the plurality of devices, the plurality of features into an aggregated feature matrix for the plurality of devices;

clustering, by applying a topic modeling algorithm to the aggregated feature matrix, the plurality of devices into a plurality of device groups according to one or more behavior groups of the plurality of device groups; and

assigning one or more devices of the plurality of devices on the network to one of the plurality of device groups to obtain an assignment.

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

updating the plurality of devices assigned to the plurality of device groups when the network traffic log is updated.

3. The method of claim **1**, wherein clustering the plurality of devices is performed hierarchically.

4. The method of claim **1**, wherein clustering the plurality of devices comprises:

selecting a traffic behavior subset of the aggregated feature matrix corresponding to traffic behavior;

applying the topic modeling algorithm to the traffic behavior subset to obtain a plurality of traffic behavior device groups;

selecting an application behavior subset of the aggregated feature matrix corresponding to application behavior;

applying, per traffic behavior device group of the plurality of traffic behavior device groups, the topic modeling algorithm to the application behavior subset to obtain a plurality of application behavior device subgroups,

wherein the plurality of device groups are the plurality of application behavior device subgroups.

5. The method of claim **1**, wherein clustering the plurality of devices comprises:

generating, by applying the topic modeling algorithm to the aggregated feature matrix, a topic feature matrix comprising a plurality of feature scores for each of a plurality of topics;

generating, by the topic modeling algorithm using the aggregated feature matrix and the topic feature matrix, a device topic matrix comprising a plurality of device topic scores for a device in the plurality of devices;

assigning, by applying a Bayesian classifier to the plurality of device topic scores of the device, a device group to the device.

6. The method of claim **1**, wherein clustering the plurality of devices comprises:

defining a particular device group of the plurality of device groups matching a plurality of behavior groups.

7. The method of claim **1**, wherein the topic modeling algorithm is a Latent Dirichlet Allocation (LDA).

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

preprocessing a plurality of log entries of the network traffic log to remove extraneous information from the plurality of log entries and obtain a preprocessed log, wherein the plurality of features are extracted from the preprocessed log.

9. The method of claim **1**, wherein aggregating the plurality of features comprises:

totaling a number of instances of an application matching an application type being associated, in the network traffic log, with a device of the plurality of devices; and

storing the number in an entry of the aggregated feature matrix matching the device and a feature corresponding to the application type.

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

generating a graphical user interface page showing the assignment; and

presenting the graphical user interface page.

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

establishing a network connection between a server group of a data intake and query system and each of one or more source network nodes, the server group comprising an indexer server and an analyzer server;

receiving source data at the server group from at least one of the one or more source network nodes via the respective network connections and transforming, by the indexer server, the source data to a plurality of timestamped entries of machine data, the plurality of timestamp entries corresponding to a plurality of log entries of the network traffic log;

storing the plurality of timestamped entries in a data store,

wherein aggregating the plurality of features comprises:

obtaining a device identifier of a device in the plurality of devices;

querying the data store for a subset of the plurality of timestamp entries that each comprise the device identifier; and

aggregate the plurality of features in the subset of the plurality of timestamp entries into a feature set for the device, the feature set being in the aggregated feature matrix.

12. The method of claim **1**, wherein the receiving the network traffic log comprises:

receiving the network traffic log of one or more devices from a firewall device.

13. The method of claim **1**, wherein the aggregating, per device of the plurality of devices, comprises:

aggregating, for a device, the plurality of features of the device into a portion of the aggregated feature matrix corresponding to the device, wherein the aggregated feature set describes one or more of an application used by the device, a destination of data sent by the device, and whether the data sent by the device crosses a firewall.

14. A system comprising:

memory comprising instructions; and

a computer processor for executing the instructions that cause the computer processor to perform operations comprising:

receiving a network traffic log of a plurality of devices in a network;

extracting, from the network traffic log, a plurality of features of the plurality of devices;

aggregating, per device of the plurality of devices, the plurality of features into an aggregated feature matrix for the plurality of devices;

clustering, by applying a topic modeling algorithm to the aggregated feature matrix, the plurality of devices into a plurality of device groups according to one or more behavior groups of the plurality of device groups; and

assigning the plurality of devices on the network to one of the plurality of device groups to obtain an assignment.

15. The system of claim **14**, wherein the operations further comprise:

updating the plurality of devices assigned to the plurality of device groups when the network traffic log is updated.

16. The system of claim **14**, wherein clustering the plurality of devices is performed hierarchically.

17. The system of claim **14**, wherein clustering the plurality of devices comprises:

selecting a traffic behavior subset of the aggregated feature matrix corresponding to traffic behavior;

applying the topic modeling algorithm to the traffic behavior subset to obtain a plurality of traffic behavior device groups;

selecting an application behavior subset of the aggregated feature matrix corresponding to application behavior;

applying, per traffic behavior device group of the plurality of traffic behavior device groups, the topic modeling algorithm to the application behavior subset to obtain a plurality of application behavior device subgroups,

wherein the plurality of device groups are the plurality of application behavior device subgroups.

18. The system of claim **14**, wherein clustering the plurality of devices comprises:

generating, by applying the topic modeling algorithm to the aggregated feature matrix, a topic feature matrix comprising a plurality of feature scores for each of a plurality of topics;

generating, by the topic modeling algorithm using the aggregated feature matrix and the topic feature matrix, a device topic matrix comprising a plurality of device topic scores for a device in the plurality of devices;

assigning, by applying a Bayesian classifier to the plurality of device topic scores of the device, a device group to the device.

19. The system of claim **14**, wherein clustering the plurality of devices comprises:

defining a particular device group of the plurality of device groups matching a plurality of behavior groups.

20. The system of claim **14**, wherein the topic modeling algorithm is a Latent Dirichlet Allocation (LDA).

21. The system of claim **14**, wherein the operations further comprise:

preprocessing a plurality of log entries of the network traffic log to remove extraneous information from the plurality of log entries and obtain a preprocessed log, wherein the plurality of features are extracted from the preprocessed log.

22. The system of claim **14**, wherein aggregating the plurality of features comprises:

totaling a number of instances of an application matching an application type being associated, in the network traffic log, with a device of the plurality of devices; and

storing the number in an entry of the aggregated feature matrix matching the device and a feature corresponding to the application type.

23. A non-transitory computer-readable storage medium storing computer-readable program code which, when executed by one or more processors, cause the one or more processors to perform operations, comprising:

receiving a network traffic log of a plurality of devices in a network;

extracting, from the network traffic log, a plurality of features of the plurality of devices;

aggregating, per device of the plurality of devices, the plurality of features into an aggregated feature matrix for the plurality of devices;

clustering, by applying a topic modeling algorithm to the aggregated feature matrix, the plurality of devices into a plurality of device groups according to one or more behavior groups of the plurality of device groups; and

assigning one or more devices of the plurality of devices on the network to one of the plurality of device groups to obtain an assignment.

24. The non-transitory computer-readable storage medium of claim **23**, the operations further comprising:

updating the plurality of devices assigned to the plurality of device groups when the network traffic log is updated.

25. The non-transitory computer-readable storage medium of claim **23**, wherein clustering the plurality of devices is performed hierarchically.

26. The non-transitory computer-readable storage medium of claim **23**, wherein clustering the plurality of devices comprises:

selecting a traffic behavior subset of the aggregated feature matrix corresponding to traffic behavior;

applying the topic modeling algorithm to the traffic behavior subset to obtain a plurality of traffic behavior device groups;

selecting an application behavior subset of the aggregated feature matrix corresponding to application behavior;

applying, per traffic behavior device group of the plurality of traffic behavior device groups, the topic modeling algorithm to the application behavior subset to obtain a plurality of application behavior device subgroups,

wherein the plurality of device groups are the plurality of application behavior device subgroups.

27. The non-transitory computer-readable storage medium of claim **23**, wherein clustering the plurality of devices comprises:

generating, by applying the topic modeling algorithm to the aggregated feature matrix, a topic feature matrix comprising a plurality of feature scores for each of a plurality of topics;

generating, by the topic modeling algorithm using the aggregated feature matrix and the topic feature matrix, a device topic matrix comprising a plurality of device topic scores for a device in the plurality of devices;

assigning, by applying a Bayesian classifier to the plurality of device topic scores of the device, a device group to the device.

28. The non-transitory computer-readable storage medium of claim **23**, wherein clustering the plurality of devices comprises:

defining a particular device group of the plurality of device groups matching a plurality of behavior groups.

29. The non-transitory computer-readable storage medium of claim **23**, wherein the topic modeling algorithm is a Latent Dirichlet Allocation (LDA).

30. The non-transitory computer-readable storage medium of claim **23**, wherein aggregating the plurality of features comprises:

totaling a number of instances of an application matching an application type being associated, in the network traffic log, with a device of the plurality of devices; and

storing the number in an entry of the aggregated feature matrix matching the device and a feature corresponding to the application type.

* * * * *