



US012392927B1

(12) **United States Patent**
Kakde et al.(10) **Patent No.:** US 12,392,927 B1
(45) **Date of Patent:** Aug. 19, 2025(54) **METHODS AND SYSTEMS FOR ALLOCATION OF FLOOD SENSORS VIA DISTRIBUTED PARAMETER FLOOD MODELING**(71) Applicant: **SAS Institute Inc.**, Cary, NC (US)(72) Inventors: **Deovrat Kakde**, Cary, NC (US); **Rajendra Singh Solanki**, Cary, NC (US); **Tyson Echentile**, Palm City, FL (US)(73) Assignee: **SAS INSTITUTE INC.**, Cary, NC (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **18/975,396**(22) Filed: **Dec. 10, 2024****Related U.S. Application Data**

(60) Provisional application No. 63/655,230, filed on Jun. 3, 2024, provisional application No. 63/642,848, filed on May 5, 2024.

(51) **Int. Cl.**

G01W 1/10 (2006.01)
G01W 1/14 (2006.01)
G06F 16/29 (2019.01)
G06F 18/23 (2023.01)

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

CPC **G01W 1/10** (2013.01); **G01W 1/14** (2013.01); **G06F 16/29** (2019.01); **G06F 18/23** (2023.01)

(58) **Field of Classification Search**

CPC G01W 1/10; G01W 1/14; G06F 16/29;
G06F 18/23

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2014/0244695 A1 * 8/2014 Battcher G06F 16/29
707/807
2019/0316309 A1 * 10/2019 Wani G06F 3/0484
2019/0318440 A1 * 10/2019 Wani G06Q 50/265
2023/0259798 A1 * 8/2023 Modugula G06Q 10/0635
706/10

(Continued)

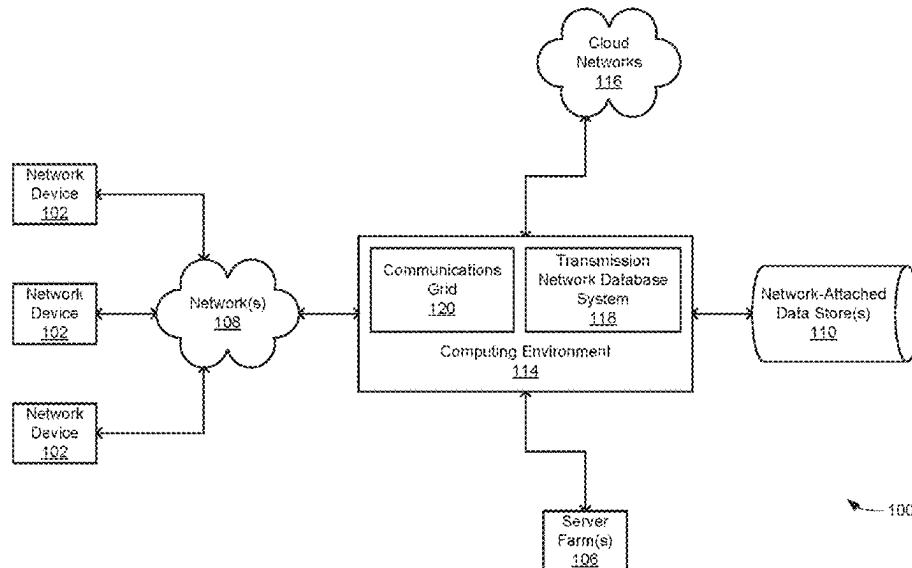
OTHER PUBLICATIONS

Floodmapp, "Enhance Your Hurricane Response with Live Mapping," Aug. 9, 2023, pp. 1-12.

(Continued)

Primary Examiner — Ricky Go*(74) Attorney, Agent, or Firm* — Padowithz Alce; Quintin Scheitlin; Alce PLLC(57) **ABSTRACT**

A computer-implemented method includes executing, via a flood simulation model, a computer simulation that simulates flooding within a target spatial area based on an input of a geospatial dataset and the rainfall intensity data of one or more flooding events associated with the target spatial area; determining normalized inundation scores for a set of geographical cells and one or more clusters of interconnected geographical cells based on executing the computer simulation; determining, via an optimization algorithm, a set of optimal sensor locations in the target spatial area based on the normalized inundation scores and the one or more clusters of interconnected geographical cells; and generating a sensor location map that includes the set of geographical cells and a set of geospatial markers identifying the set of sensor locations; and generating a visual output that displays the sensor location map.

30 Claims, 25 Drawing Sheets

(56)

References Cited

U.S. PATENT DOCUMENTS

2024/0242289 A1* 7/2024 Jiang G06Q 40/08

OTHER PUBLICATIONS

Floodmapp, "FloodMapp Assists Vermont Flooding Response and Recovery," Jul. 18, 2023, pp. 1-13.

Ogie et al., "Optimal placement of water-level sensors to facilitate data-driven management of hydrological infrastructure assets in coastal mega-cities of developing nations," Elsevier, 2017, pp. 385-395.

Wang et al., "Urban flood prediction under heavy precipitation," Elsevier, 2019, pp. 1-21.

* cited by examiner

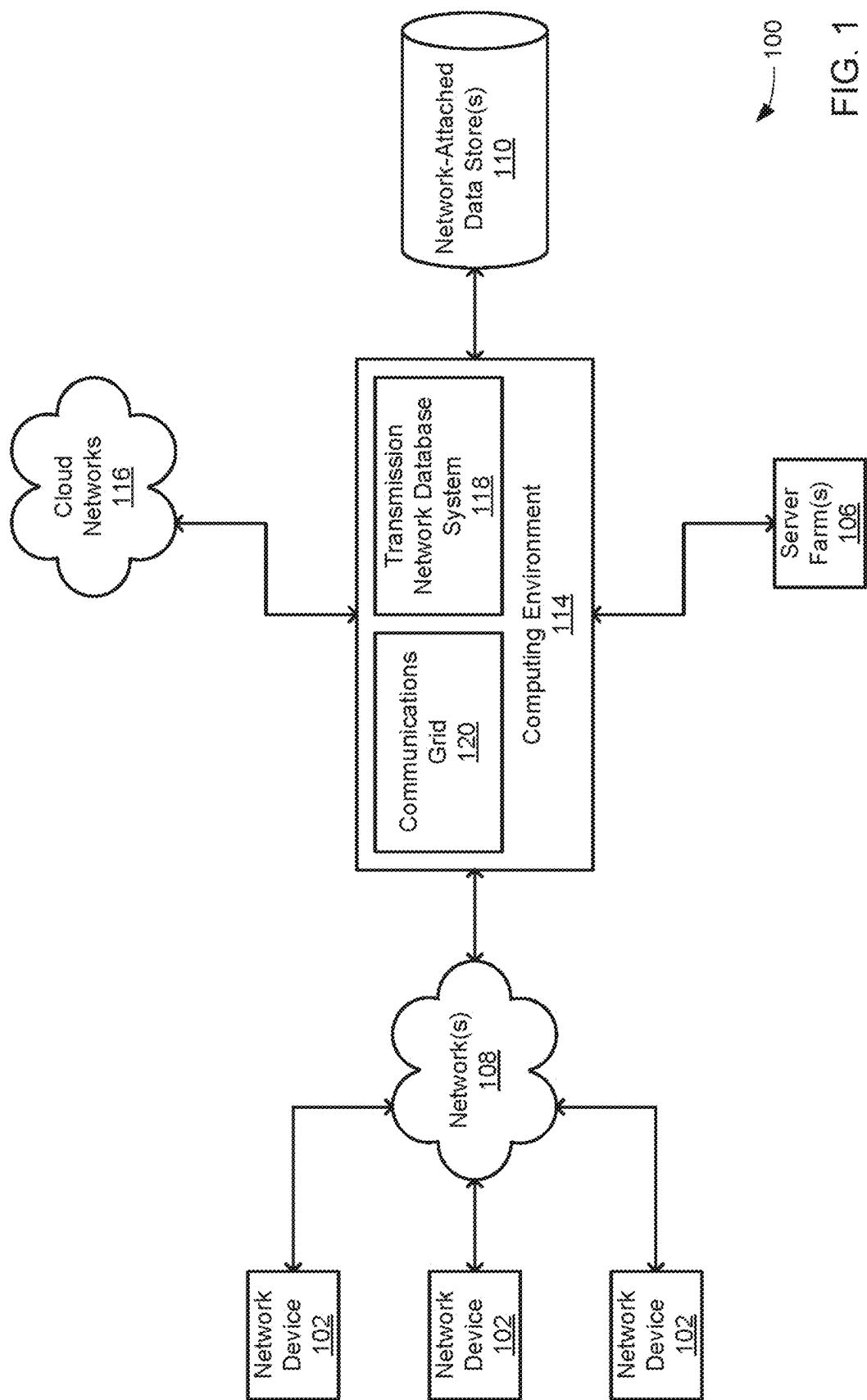


FIG. 1

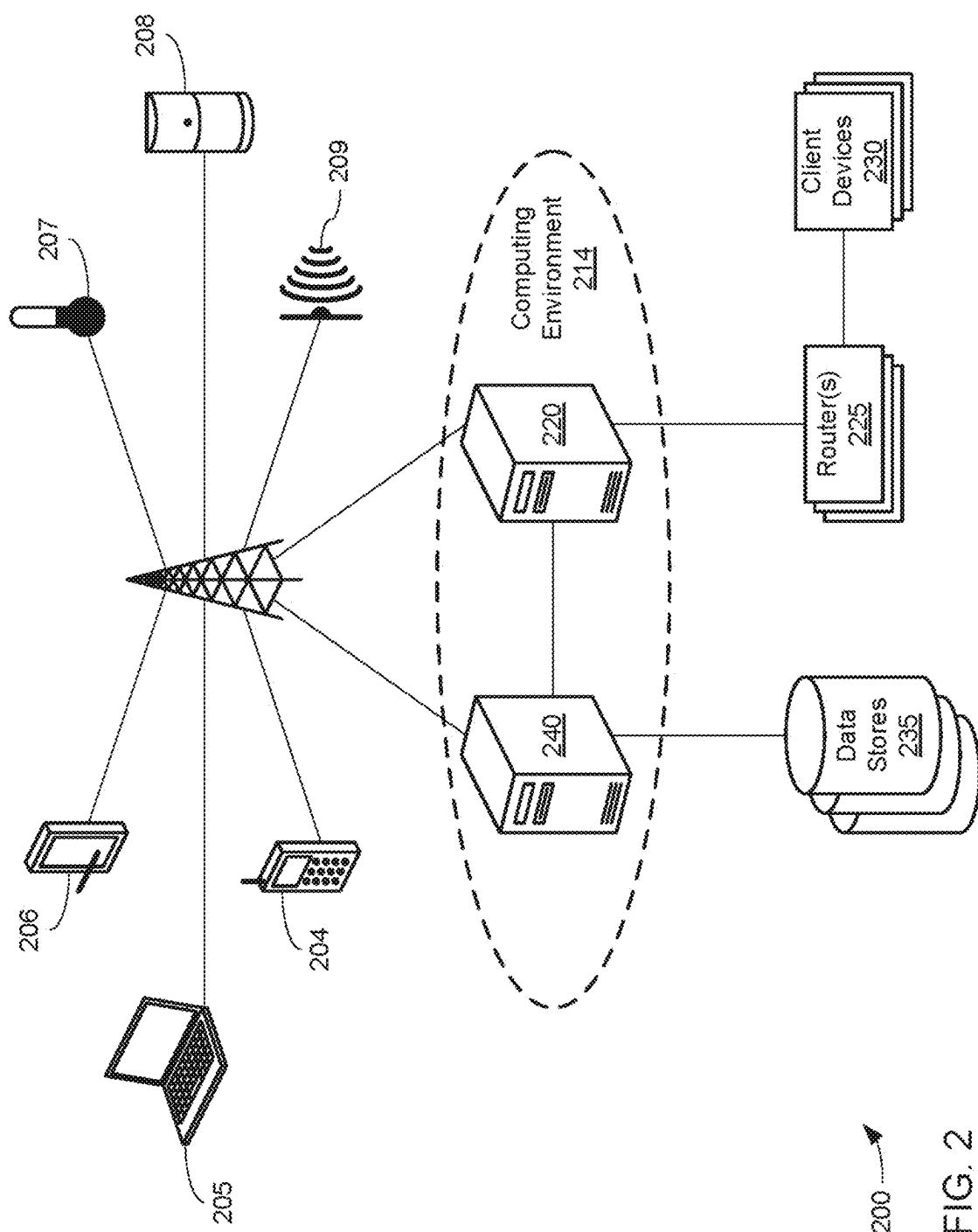
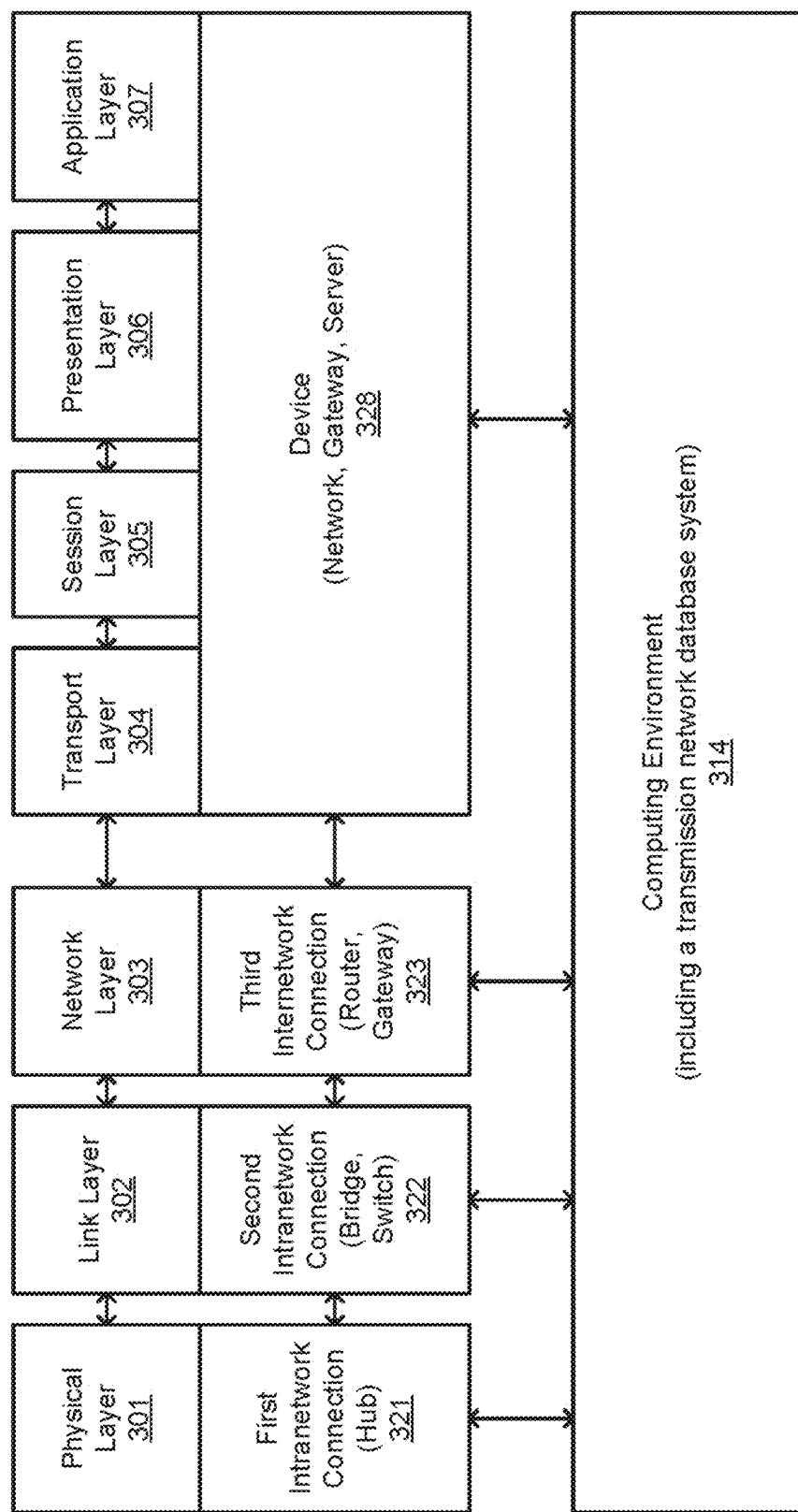


FIG. 2



300 —►

FIG. 3

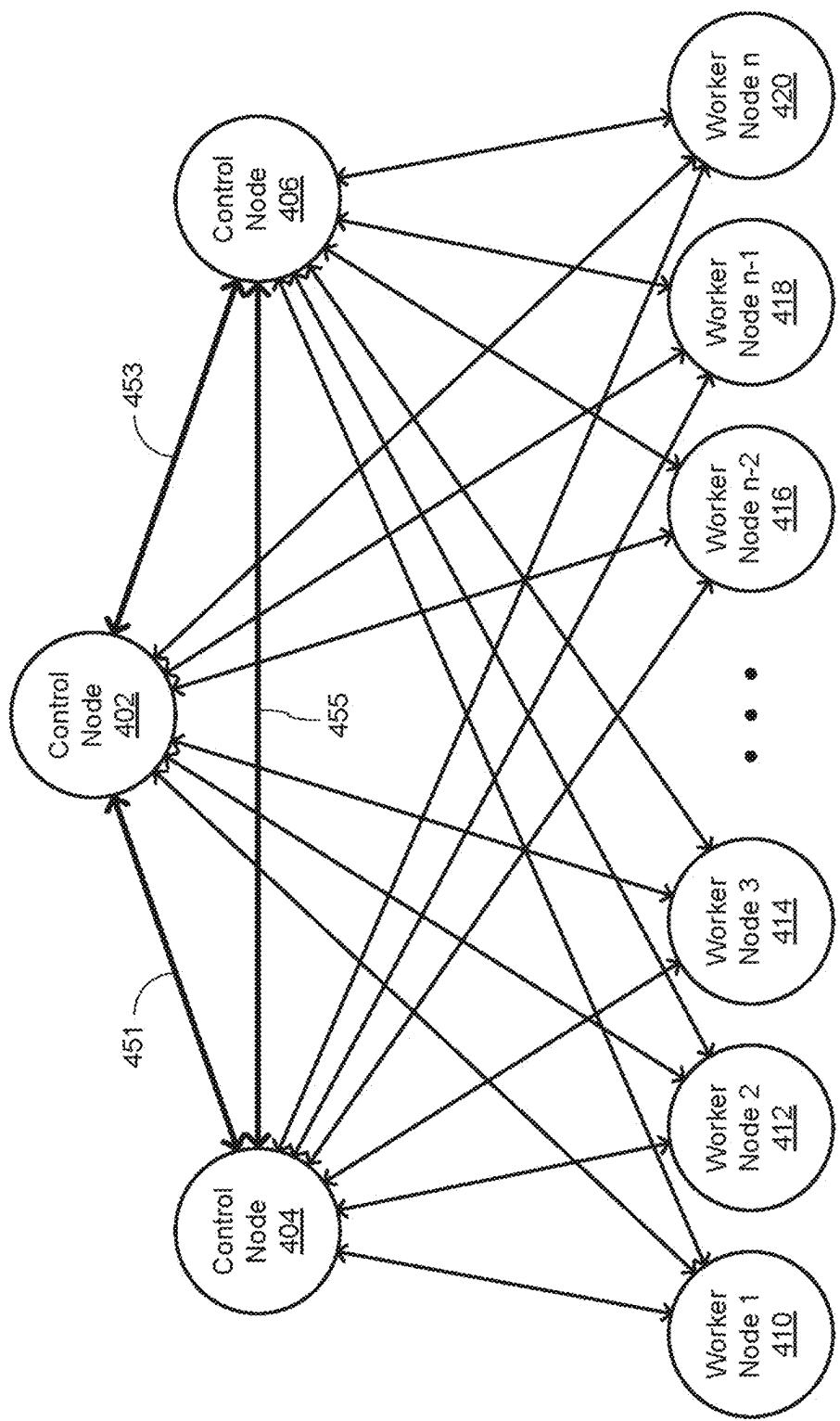
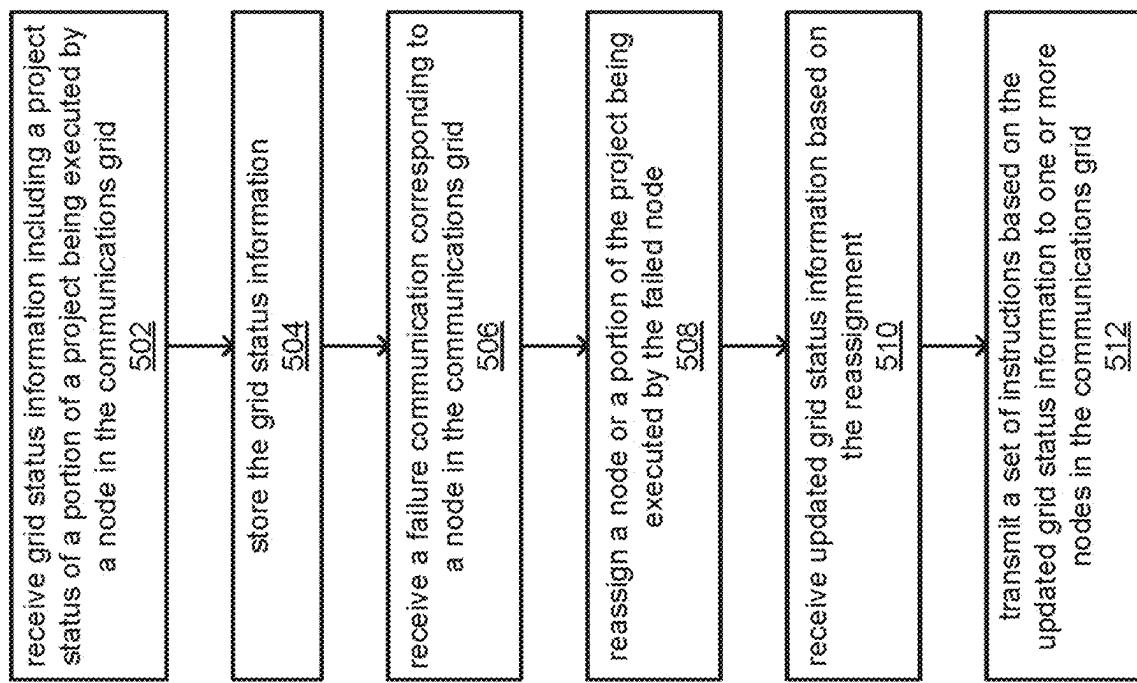


FIG. 4



500 →

FIG. 5

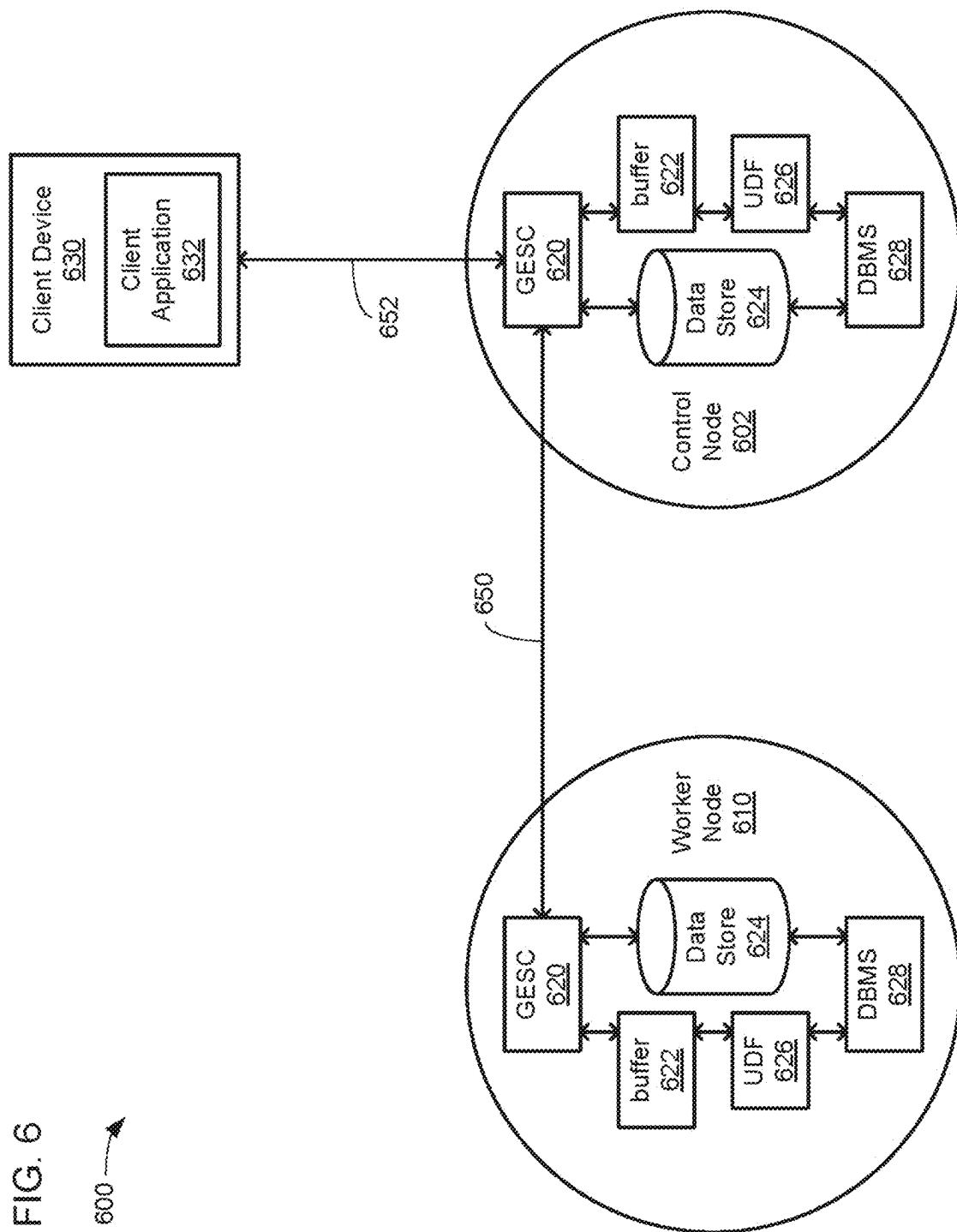
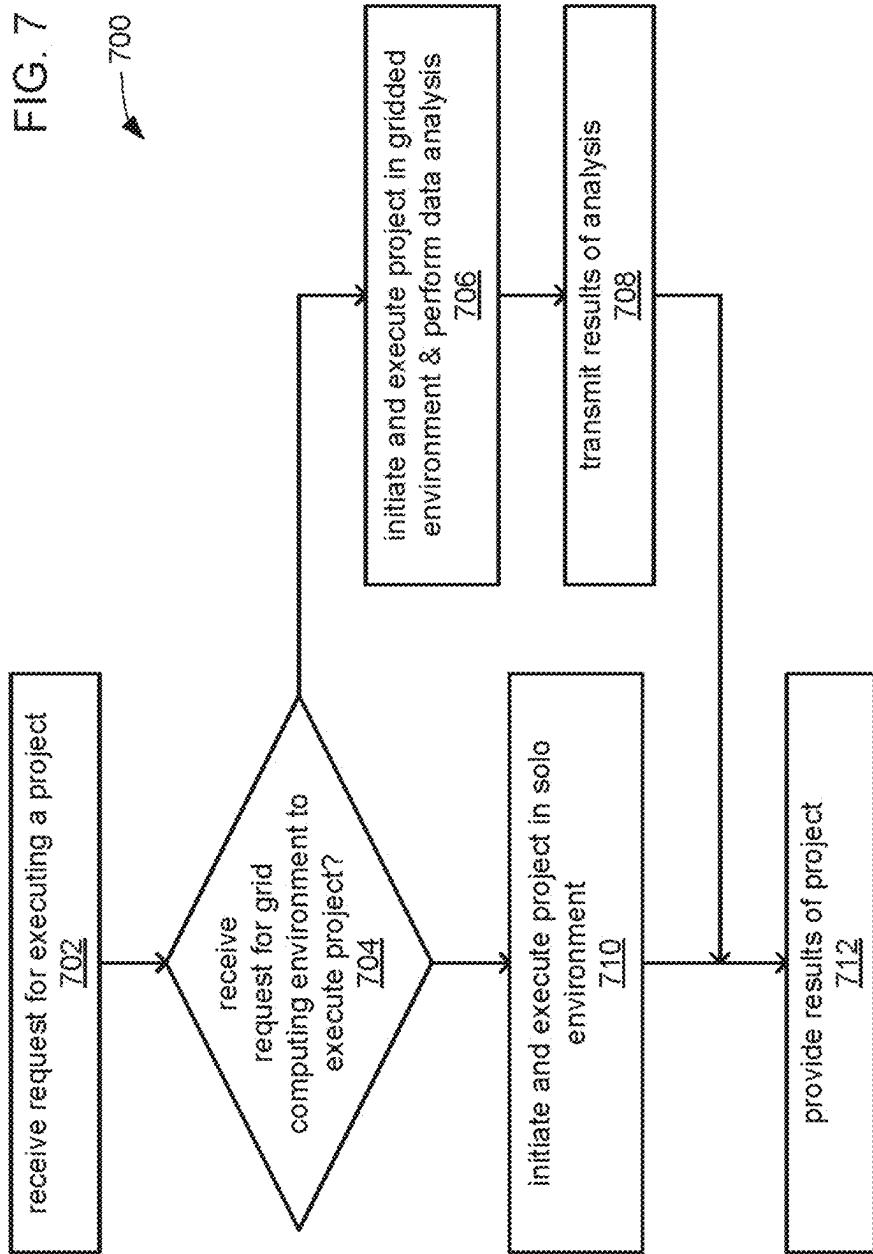


FIG. 7



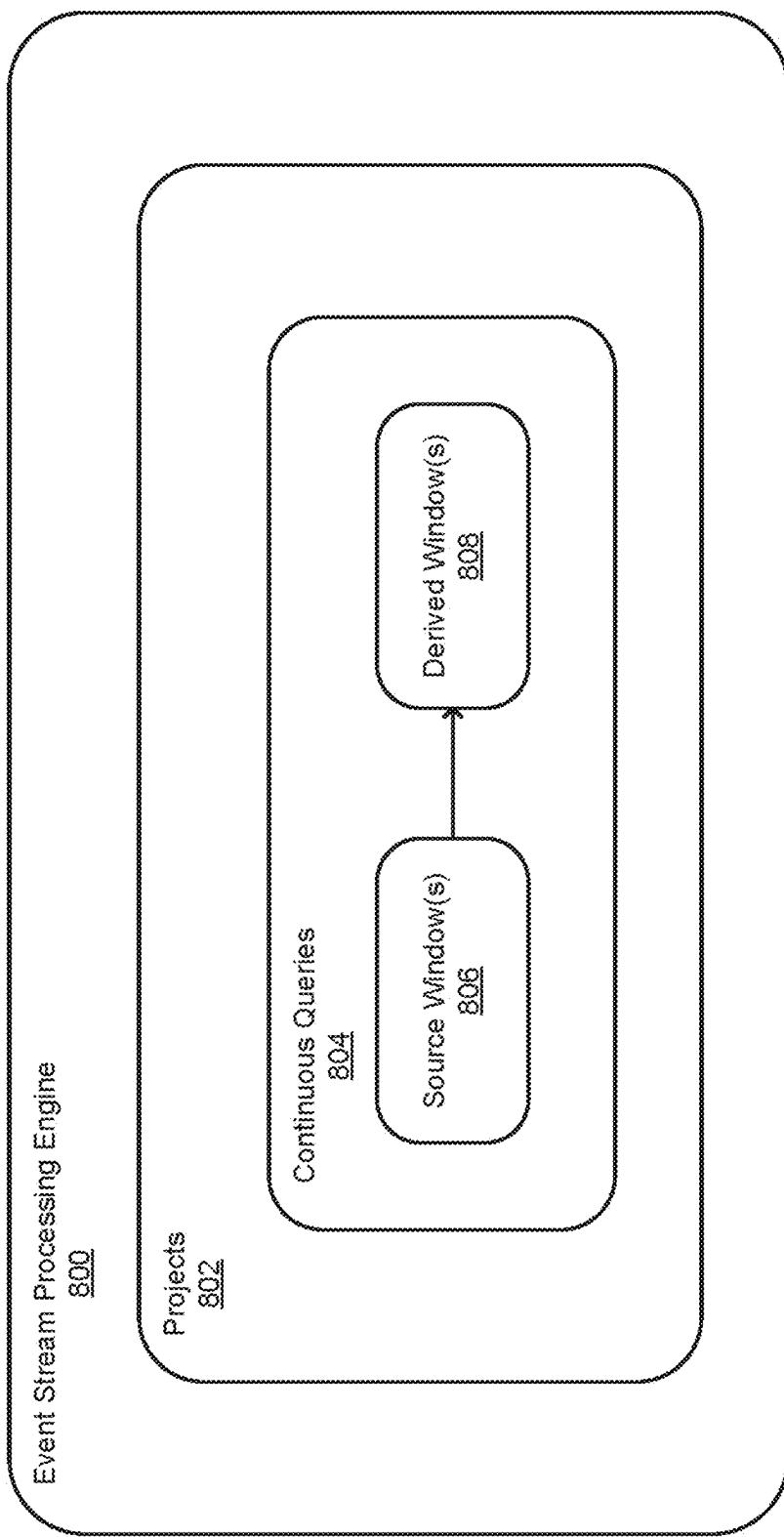
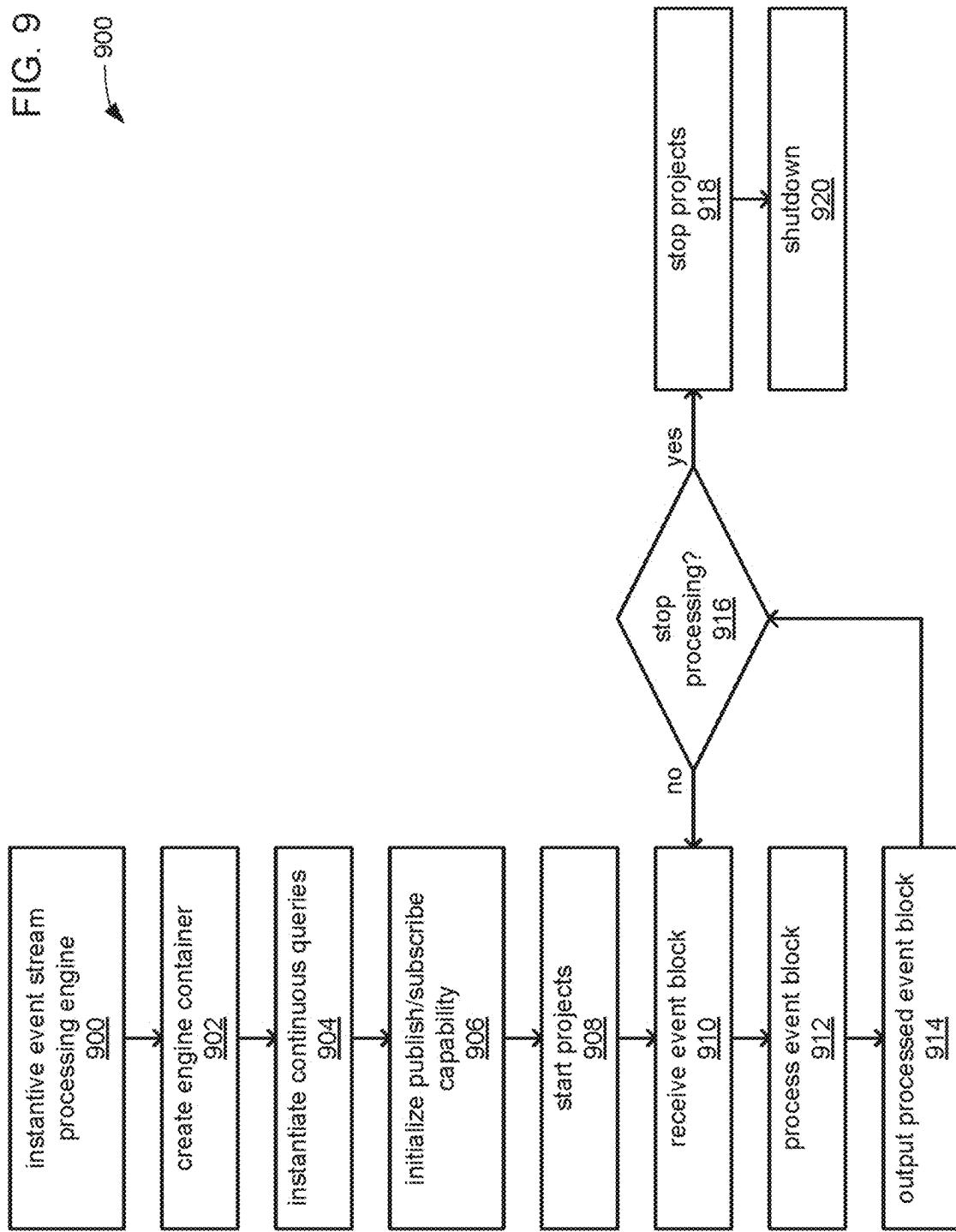
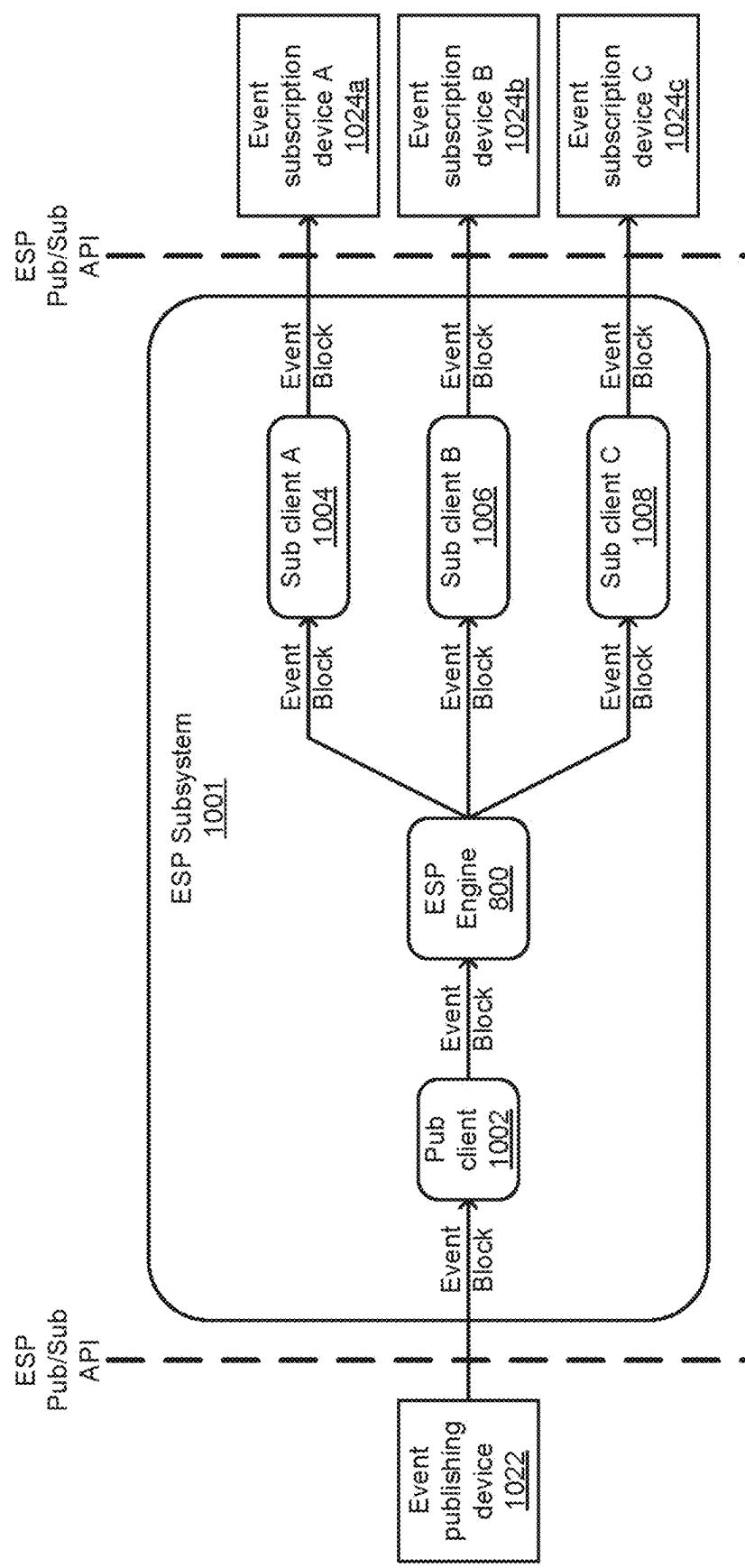


FIG. 8

FIG. 9





1000

FIG. 10

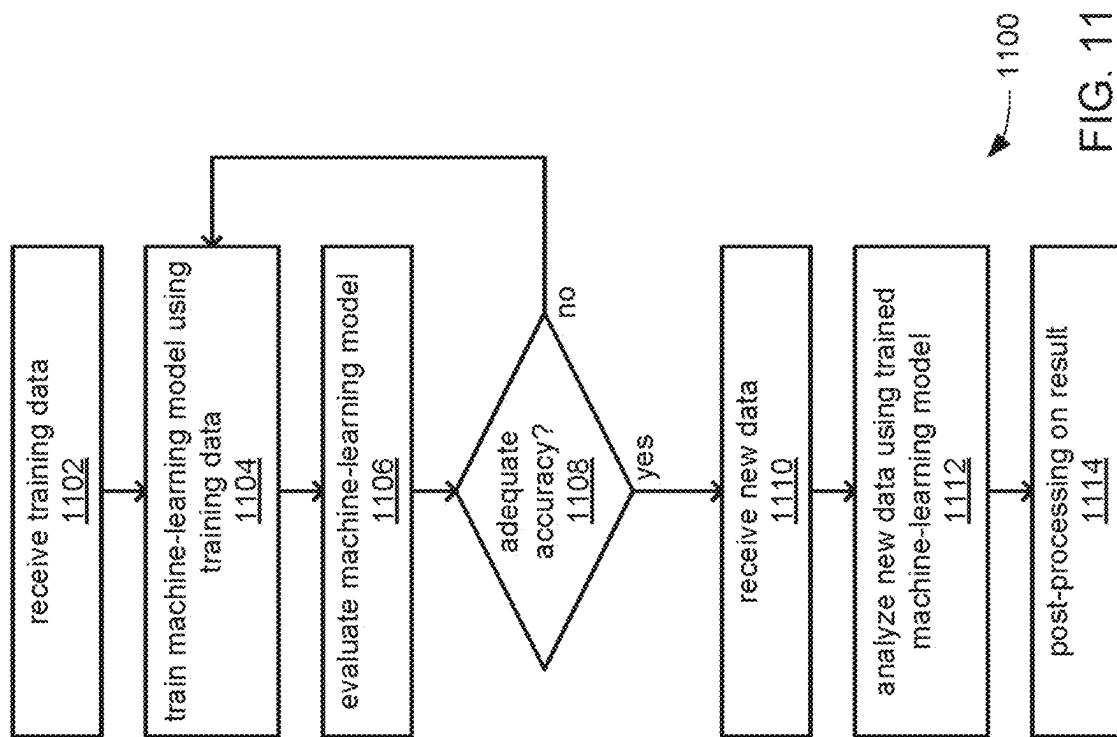
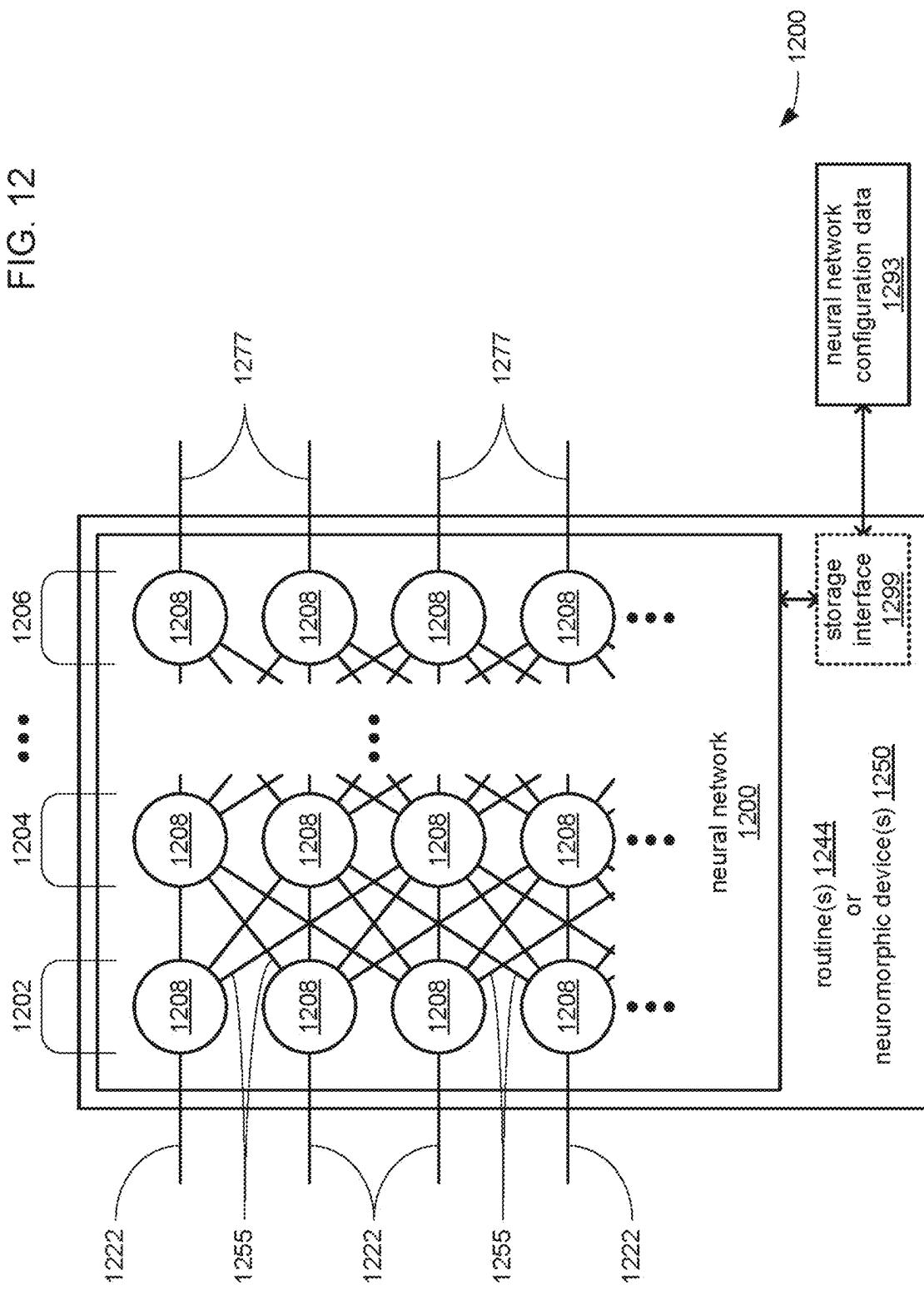
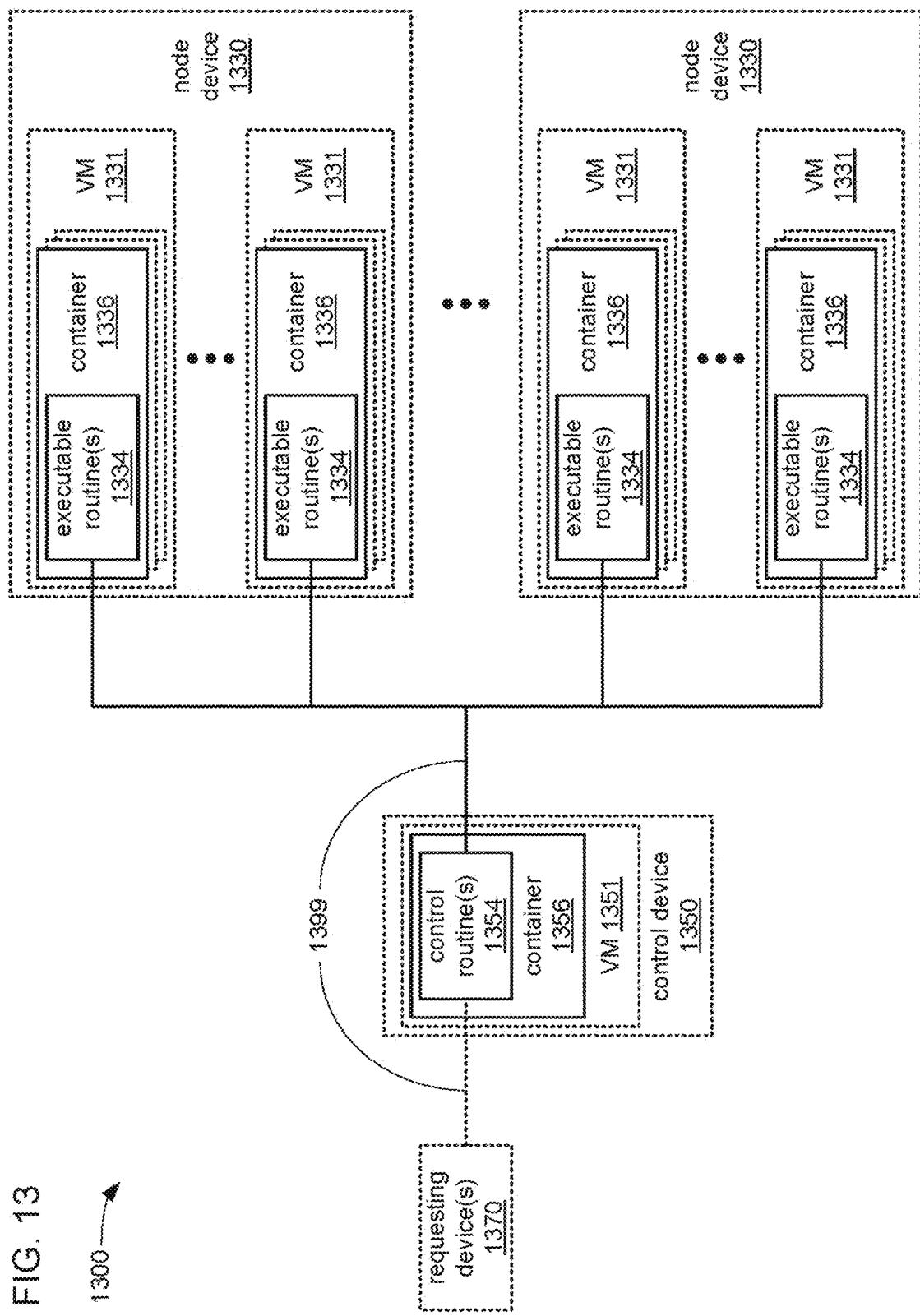


FIG. 11

FIG. 12





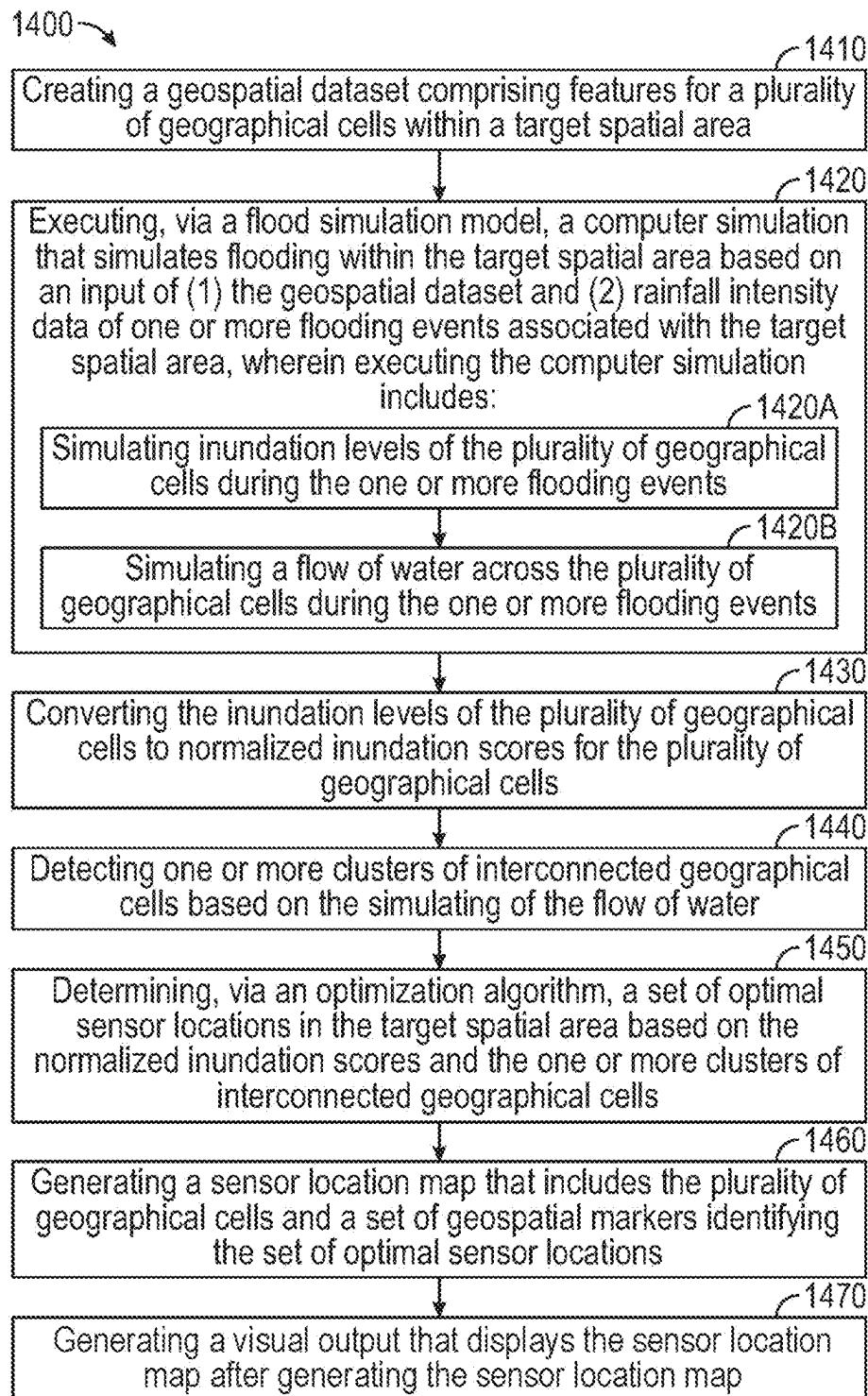


FIG. 14

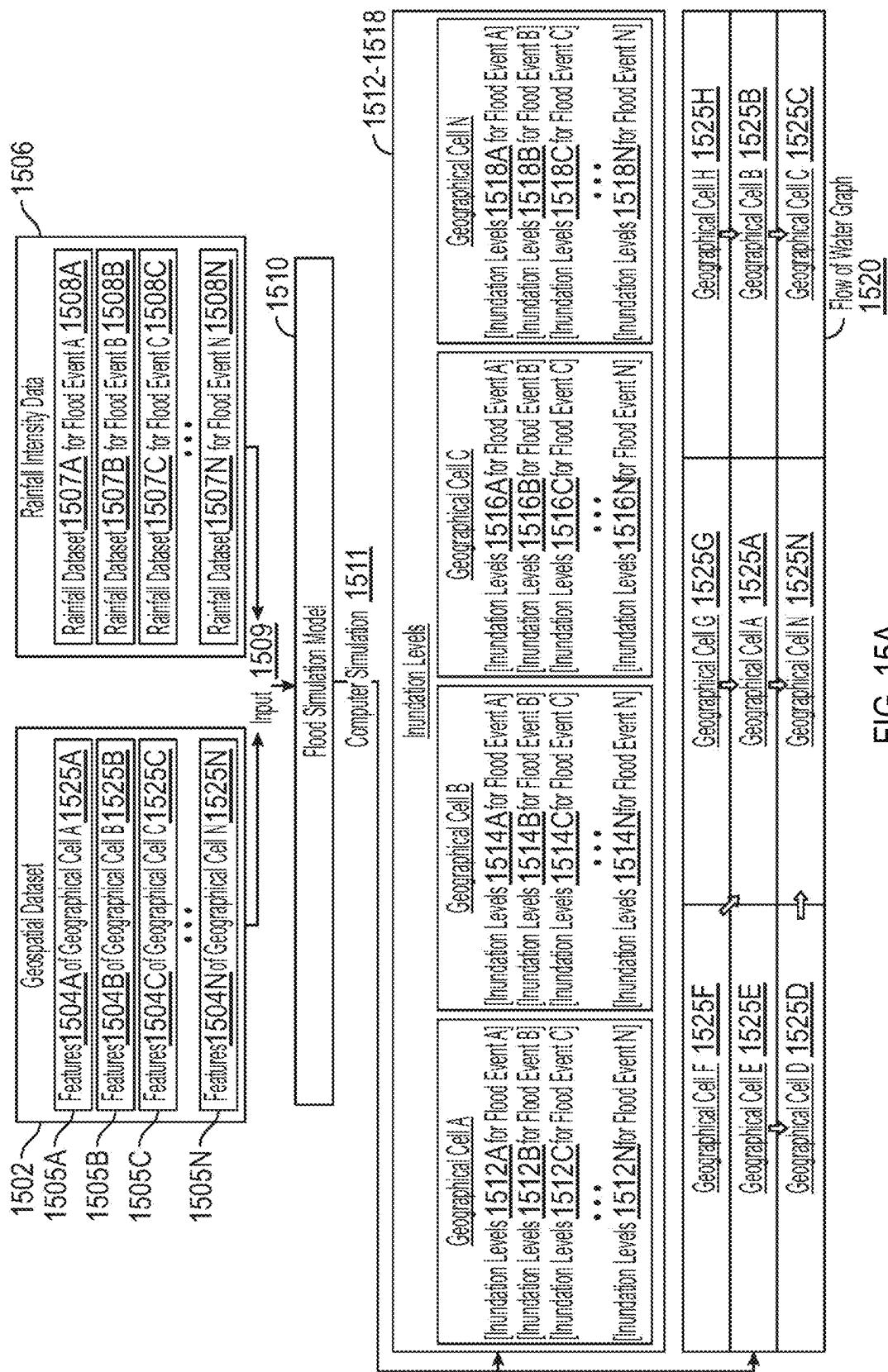


FIG. 15A

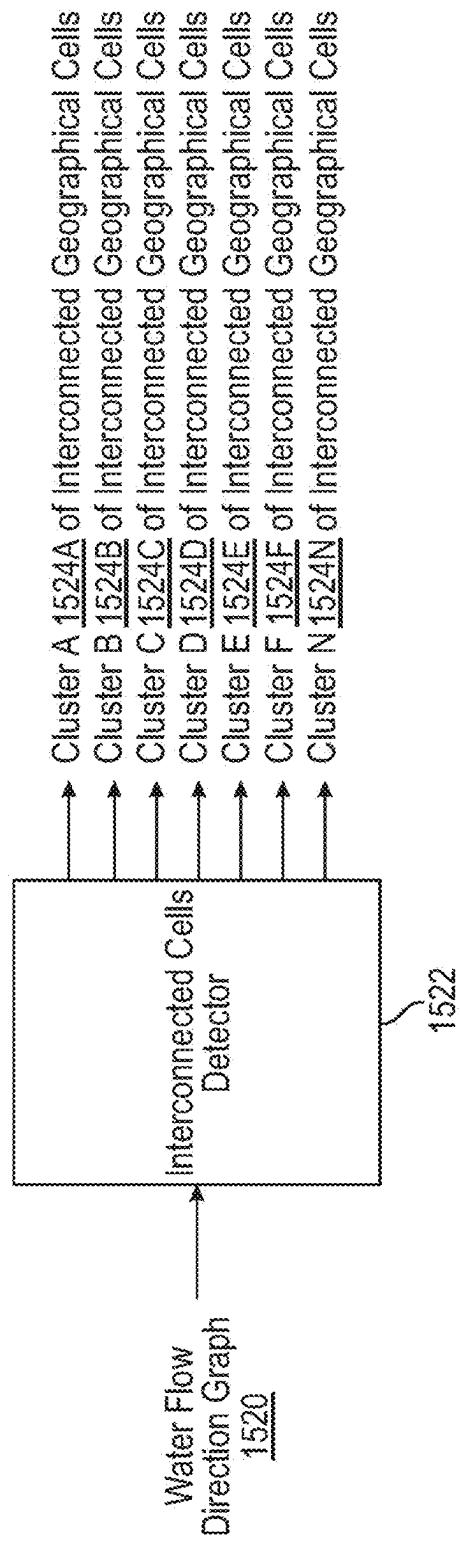


FIG. 15B

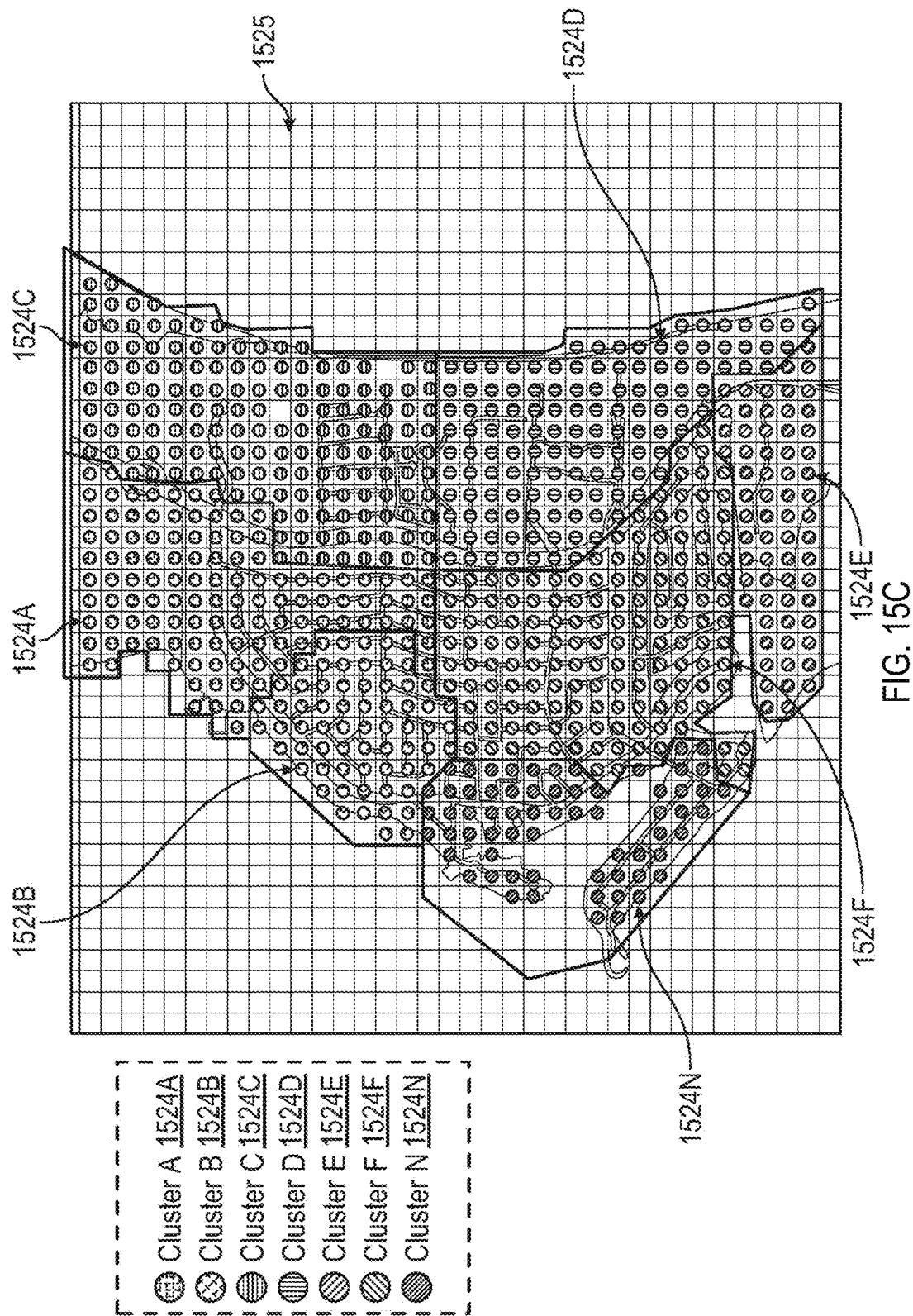


FIG. 15C

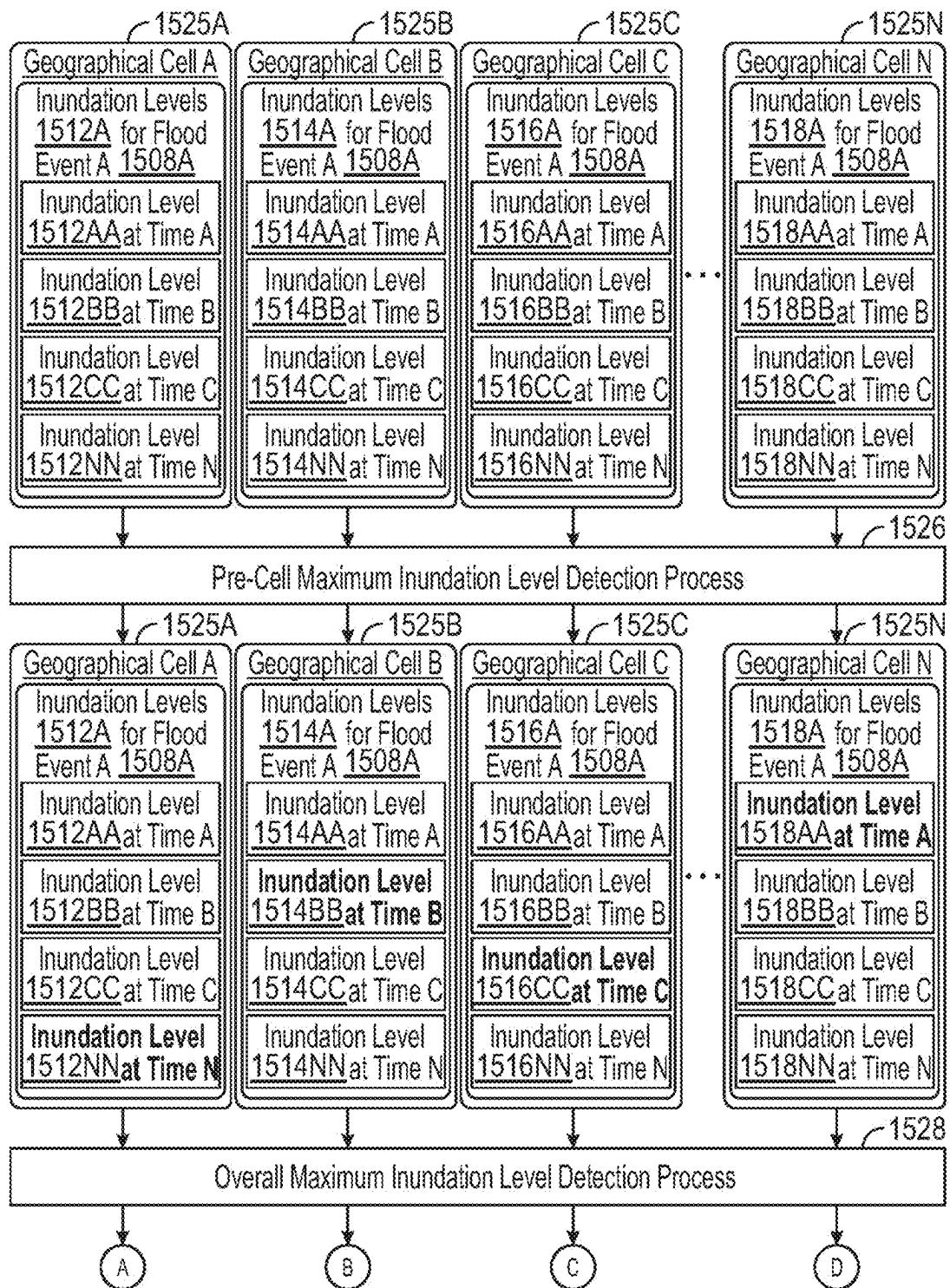


FIG. 15D-1

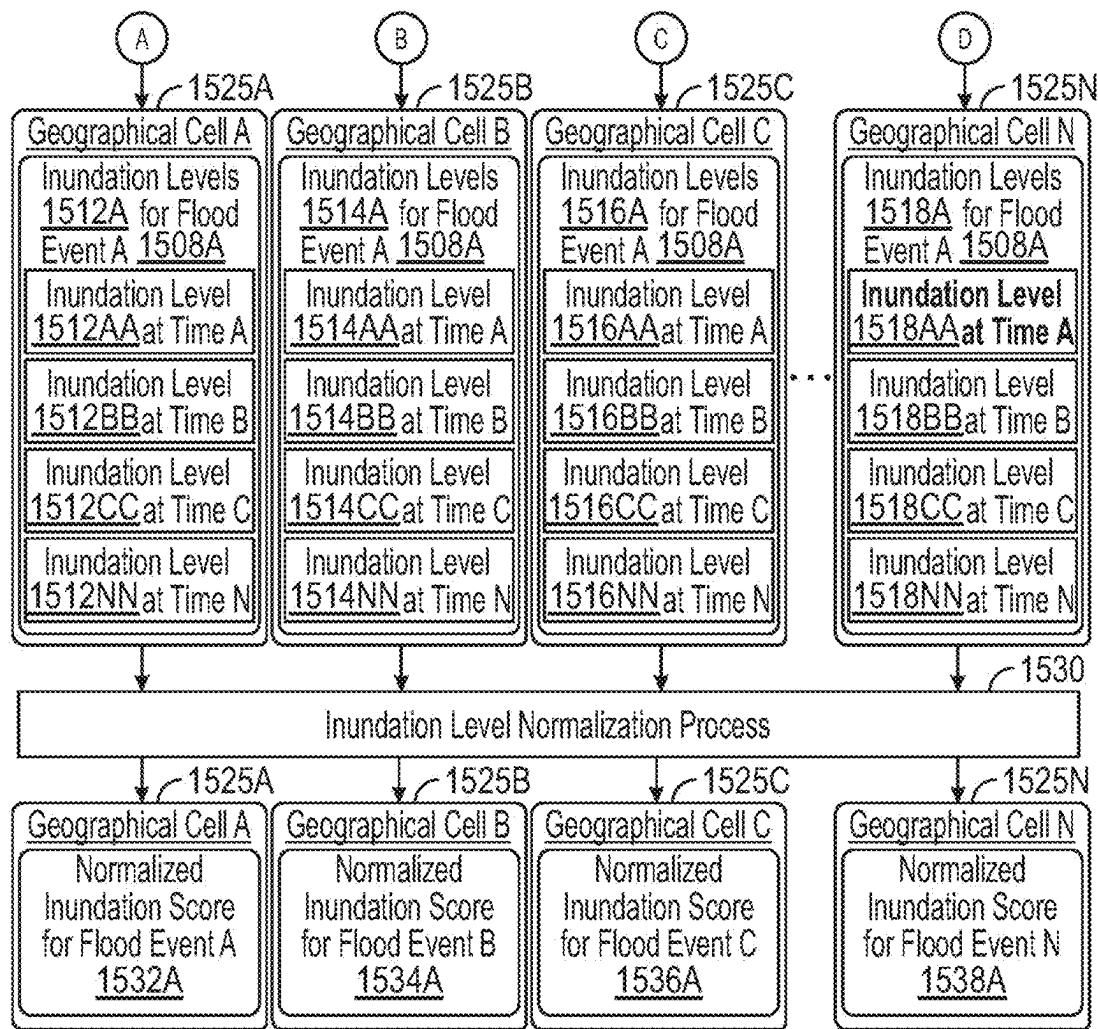


FIG. 15D-2

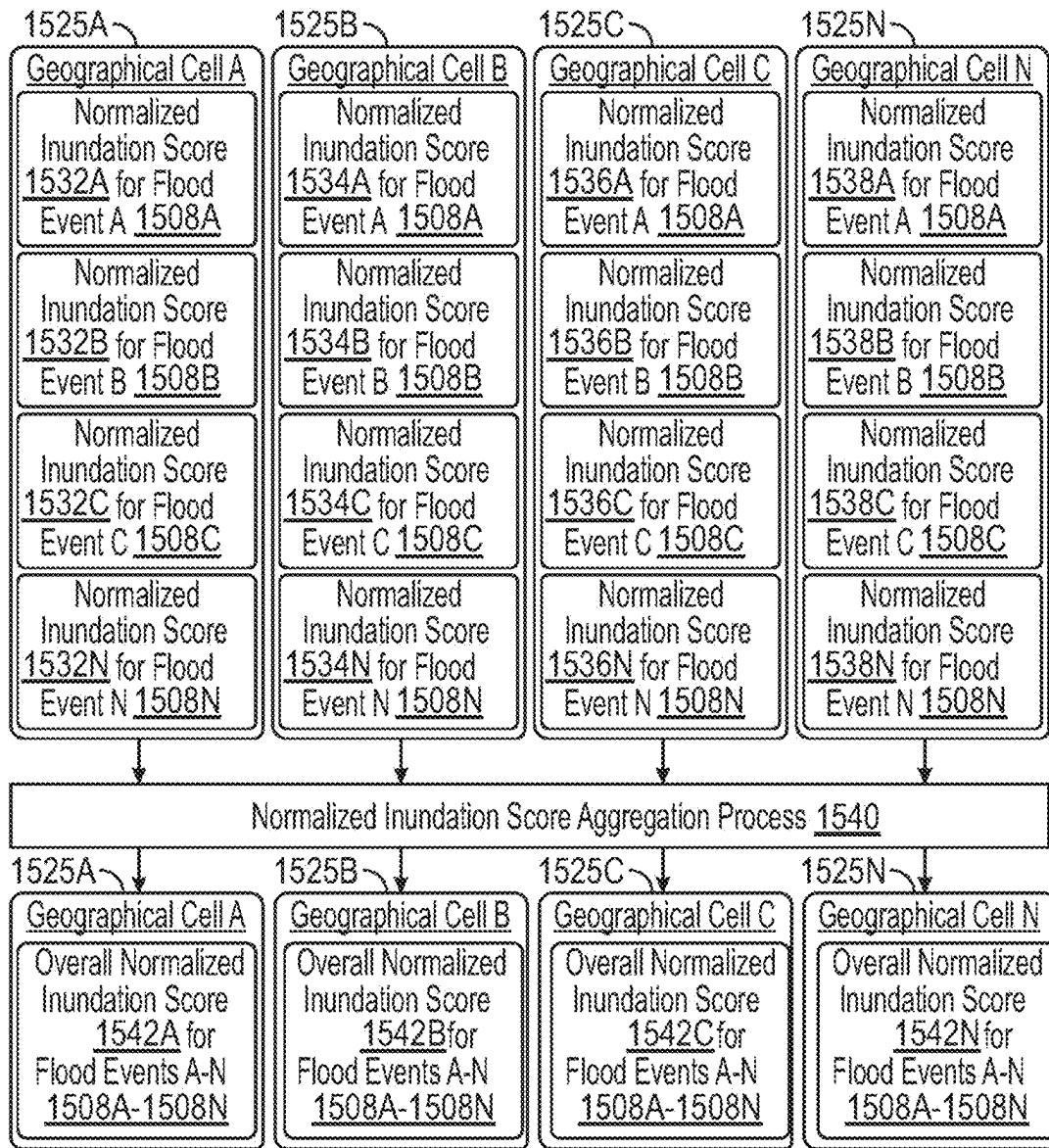


FIG. 15E

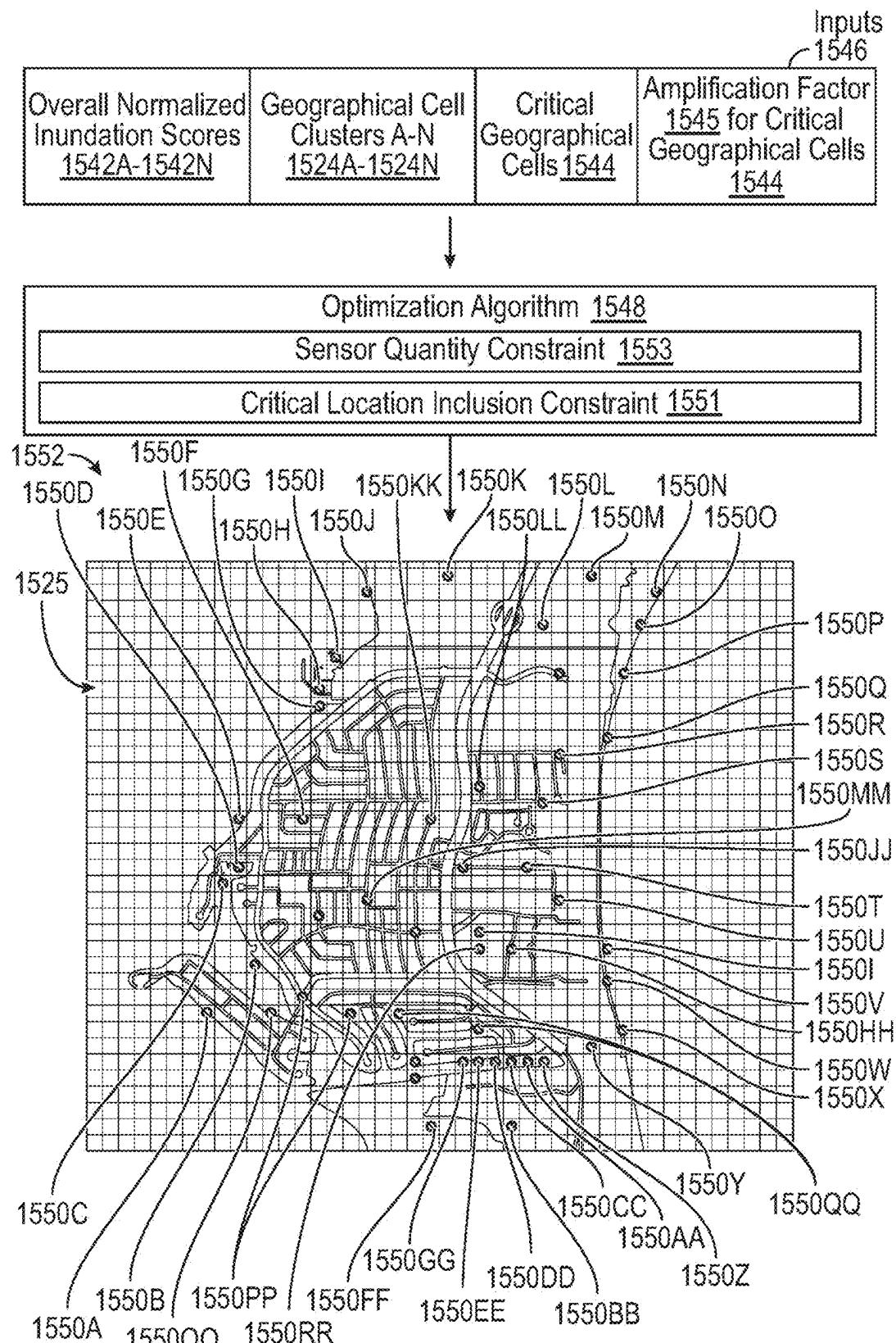
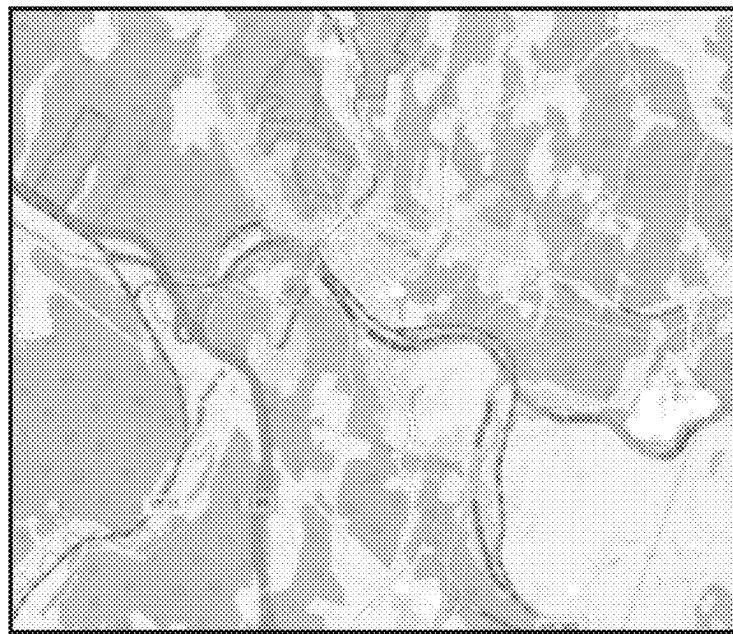
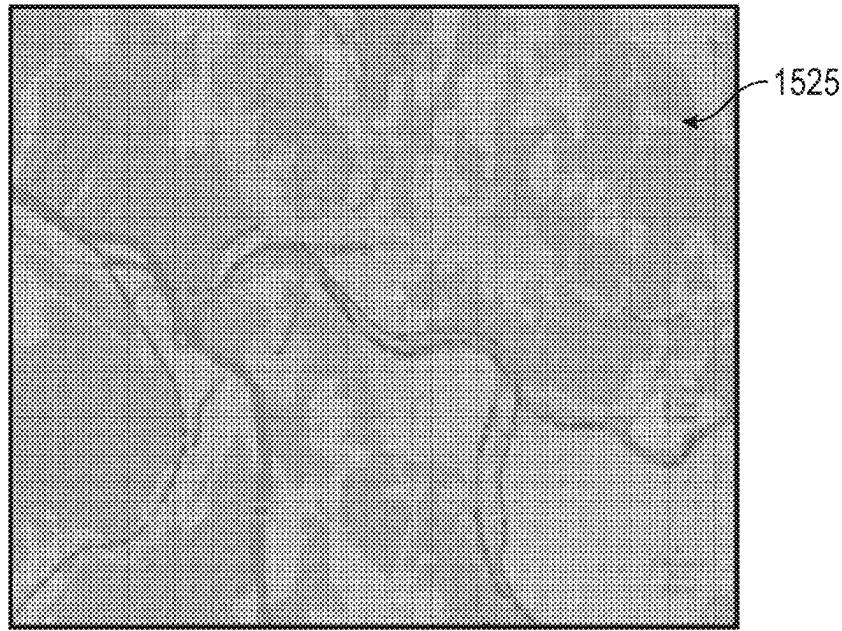


FIG. 15F



Representantion 1554 of a Target Spatial Area



Grid 1556

FIG. 15G-1

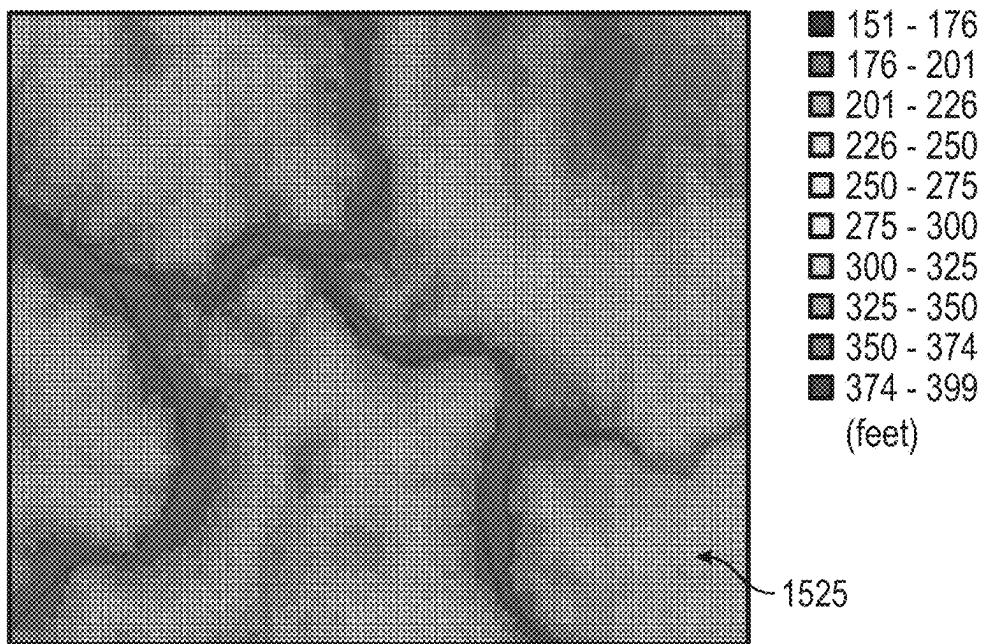
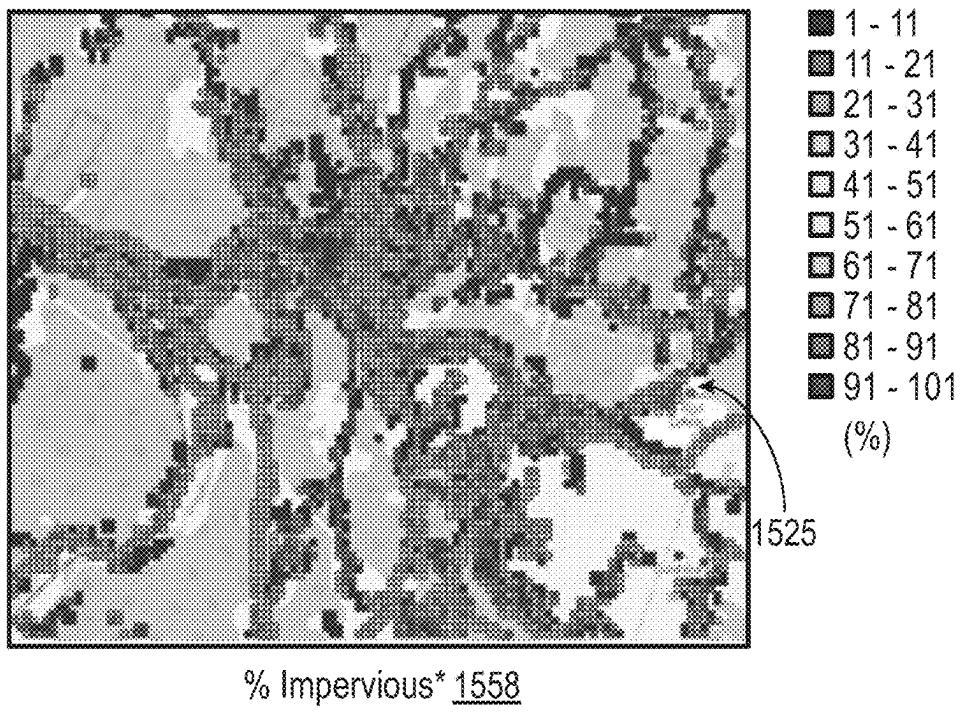


FIG. 15G-2

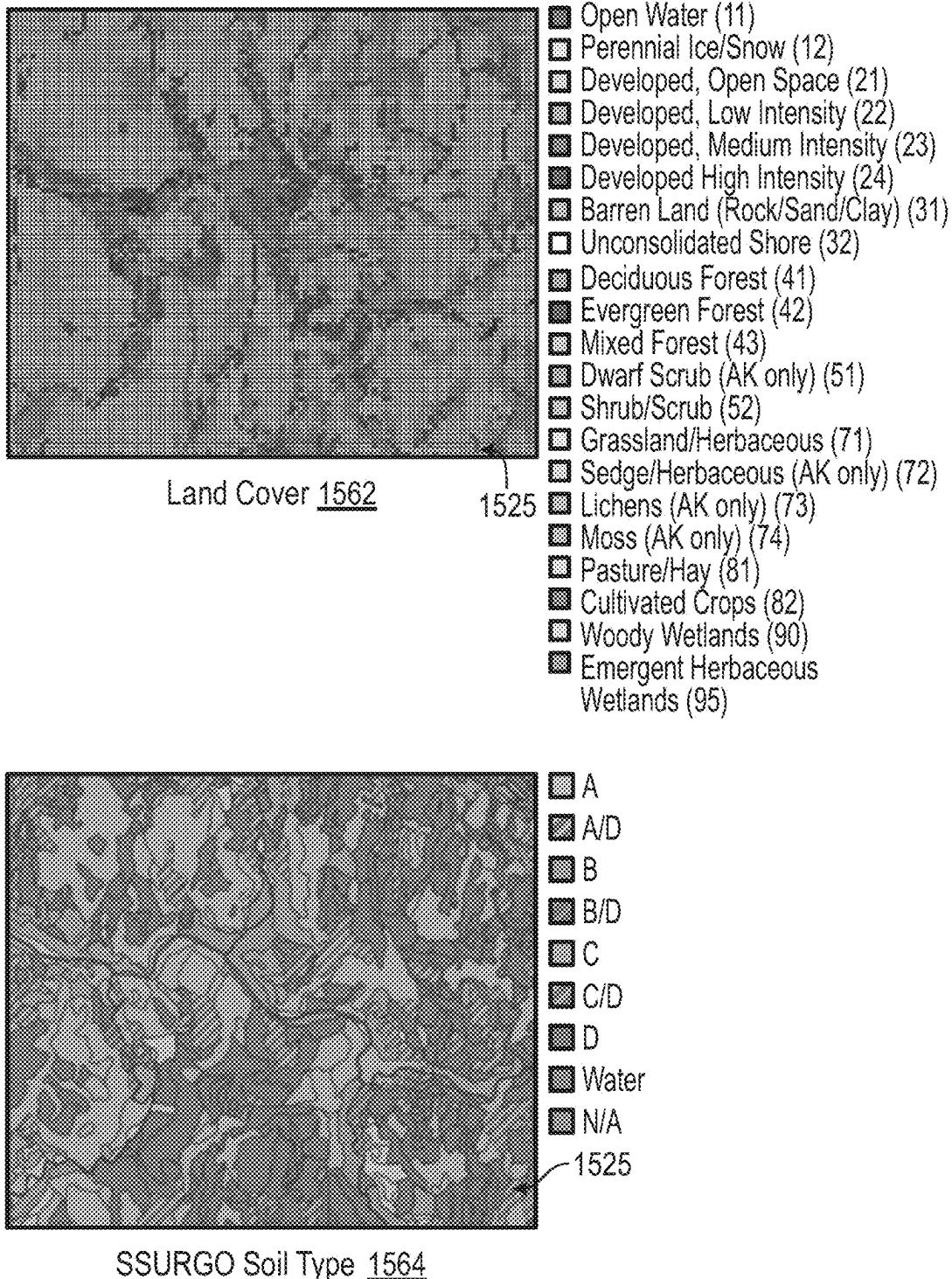


FIG. 15G-3

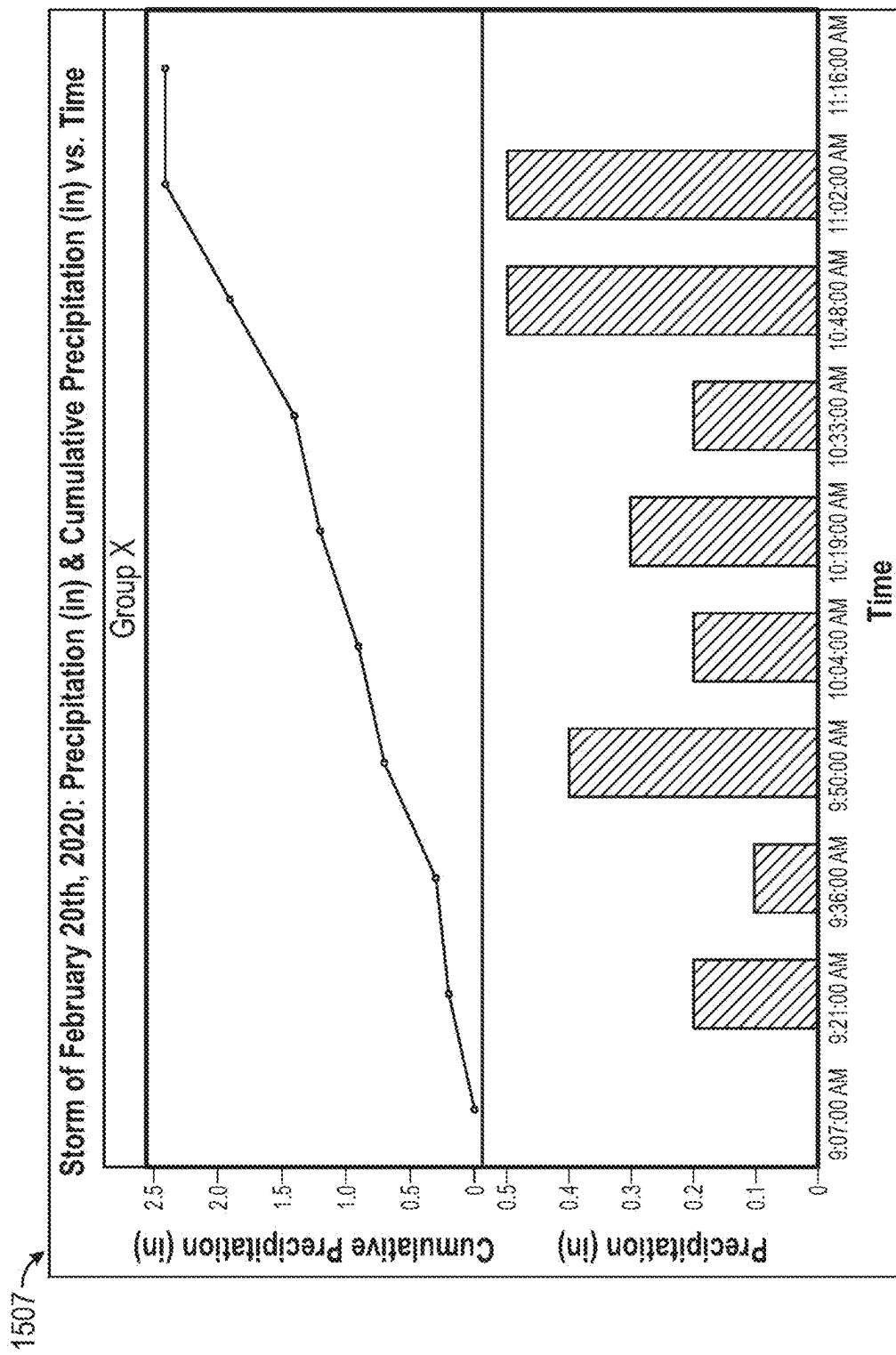


FIG. 15H

1

**METHODS AND SYSTEMS FOR
ALLOCATION OF FLOOD SENSORS VIA
DISTRIBUTED PARAMETER FLOOD
MODELING**

**CROSS-REFERENCE TO RELATED
APPLICATIONS**

This application claims the benefit of U.S. Provisional Application No. 63/642,848, filed on 5 May 2024, and U.S. Provisional Application No. 63/655,230, filed on 3 Jun. 2024, which are incorporated in their entirety by this reference.

TECHNICAL FIELD

This invention relates generally to flood modeling and, more specifically, to allocation of flood sensors via distributed parameter flood modeling.

BACKGROUND

Flooding may pose risks to infrastructure, public safety, and the environment. In order to monitor and assess the impact of floods, flood sensors may be placed within a region where flooding may occur. However, the number of flood sensors that may be placed and/or monitored at a given time may be limited. Accordingly, flooding prediction models may be used to determine where the limited flood sensors may be placed to best cover flooding-related objectives.

Conventional systems that perform flood prediction and mitigation may utilize models that rely on the presence of bodies of water, such as rivers or lakes. However, such systems may lack the capability to address instances of flooding that occur in regions without water bodies (e.g., flash flooding in desert regions). Accordingly, these systems may fail to provide accurate flood predictions when tasked to model regions that do not have bodies of water and may accordingly fail to provide information that may enable effective placement of flood sensors.

Additionally, conventional models may fail to account for regions with high levels of impervious surfaces (e.g., concrete and asphalt), such as regions in which urban flooding occurs. However, such regions may be associated with reduced water infiltration and increased surface runoff. Failing to account for these impervious surfaces may result in predictions that may underestimate the effect of water flow in these regions. The techniques described herein may enable determination of where sensors are to be placed in regions without permanent bodies and may further account for the effects of impervious surfaces on water flow.

BRIEF SUMMARY OF THE EMBODIMENTS(S)

This summary is not intended to identify only key or essential features of the described subject matter, nor is it intended to be used in isolation to determine the scope of the described subject matter. The subject matter should be understood by reference to appropriate portions of the entire specification of this patent, any or all drawings, and each claim.

In some embodiments, a computer-program product may comprise a non-transitory machine-readable storage medium storing computer instructions that, when executed by one or more processors, perform operations comprising: creating a geospatial dataset comprising features for a plurality of geographical cells within a target spatial area;

2

executing, via a flood simulation model, a computer simulation that simulates flooding within the target spatial area based on an input of (1) the geospatial dataset and (2) rainfall intensity data of one or more flooding events associated with the target spatial area, wherein executing the computer simulation includes: simulating inundation levels of the plurality of geographical cells during the one or more flooding events, and simulating a flow of water across the plurality of geographical cells during the one or more flooding events; converting the inundation levels of the plurality of geographical cells to normalized inundation scores for the plurality of geographical cells; detecting one or more clusters of interconnected geographical cells based on the simulating of the flow of water; determining, via an optimization algorithm, a set of optimal sensor locations in the target spatial area based on the normalized inundation scores and the one or more clusters of interconnected geographical cells; generating a sensor location map that includes the plurality of geographical cells and a set of geospatial markers identifying the set of optimal sensor locations; and generating a visual output that displays the sensor location map after generating the sensor location map.

In some embodiments of the computer-program product, simulating the inundation levels of the plurality of geographical cells during a respective flooding event includes: identifying, from the rainfall intensity data, precipitation amounts observed during a plurality of timesteps of the respective flooding event, simulating, via the flood simulation model, the precipitation amounts observed during the plurality of timesteps on the plurality of geographical cells, and determining maximum inundation levels of the plurality of geographical cells based at least on the simulating of the precipitation amounts.

In some embodiments of the computer-program product, determining a maximum inundation level of a respective geographical cell includes: identifying an inundation level of the respective geographical cell at a first timestep of the respective flooding event, identifying inundation levels of the respective geographical cell at a plurality of other timesteps of the respective flooding event, and using the inundation level identified at the first timestep as the maximum inundation level if the inundation level identified at the first timestep is greater than the inundation levels identified at the plurality of other timesteps.

In some embodiments of the computer-program product, the inundation levels of the plurality of geographical cells include maximum inundation levels of the plurality of geographical cells during a respective flooding event, and converting an inundation level of a respective geographical cell to a normalized inundation score for the respective geographical cell includes: identifying an overall maximum inundation level among the maximum inundation levels, and computing the normalized inundation score by dividing the inundation level of the respective geographical cell by the overall maximum inundation level.

In some embodiments of the computer-program product, simulating an inundation level of a respective geographical cell includes: identifying precipitation amounts observed during a plurality of timesteps of a respective flooding event, simulating the precipitation amounts observed during the plurality of timesteps on the respective geographical cell, computing inundation levels of the respective geographical cell across the plurality of timesteps, and using a maximum inundation level of the inundation levels computed across the plurality of timesteps as the inundation level of the respective geographical cell.

In some embodiments of the computer-program product, computing an inundation level of a respective geographical cell at a respective timestep includes: (a) identifying a precipitation amount observed during the respective timestep, (b) identifying an amount of water flowing from other geographical cells into the respective geographical cell during the respective timestep, (c) identifying an amount of water flowing from the respective geographical cell to the other geographical cells during the respective timestep, (d) identifying an amount of water lost due to infiltration during the respective timestep, (e) identifying an amount of water removed via storm drain outflow during the respective timestep, and (f) computing the inundation level of the respective geographical cell using (a)-(e).

In some embodiments of the computer-program product, computing the inundation level of the respective geographical cell using (a)-(e) includes: summing the precipitation amount identified in (a) with the amount of water identified in (b), and subtracting the amount of water identified in (c), the amount of water identified in (d), and the amount of water identified in (e) from a result of the summing.

In some embodiments of the computer-program product, the flow of water across the plurality of geographical cells is represented using a graph data structure, and the graph data structure includes: a plurality of nodes corresponding to the plurality of geographical cells, and a plurality of edges specifying water flow trajectories among the plurality of nodes.

In some embodiments of the computer-program product, a respective edge extending from a first respective node to a second respective node specifies a water flow trajectory that indicates water moves from a geographical cell corresponding to the first respective node to a geographical cell corresponding to the second respective node during flooding.

In some embodiments of the computer-program product, the flow of water across the plurality of geographical cells is represented using a graph data structure, and detecting the one or more clusters of interconnected geographical cells includes: identifying that a first plurality of nodes of the graph data structure are connected via a first plurality of edges; and detecting the first plurality of nodes as a first cluster of interconnected geographical cells based on identifying that the first plurality of nodes are connected via the first plurality of edges.

In some embodiments of the computer-program product, detecting the one or more clusters of interconnected geographical cells further includes: identifying that a second plurality of nodes of the graph data structure are connected via a second plurality of edges; adding the second plurality of nodes to the first cluster of interconnected geographical cells if there is at least one edge connecting a node from the first plurality of nodes to a node of the second plurality of nodes, and detecting the second plurality of nodes as a second cluster of interconnected geographical cells if there is not at least one edge connecting the node from the first plurality of nodes to the node of the second plurality of nodes.

In some embodiments of the computer-program product, a respective cluster of interconnected geographical cells corresponds to a respective sub-catchment within the target spatial area, indicates flow characteristics of the respective sub-catchment, and indicates a retention size of the respective sub-catchment.

In some embodiments of the computer-program product, the optimization algorithm maximizes a multi-criterion objective function to determine the set of optimal sensor

locations, and the multi-criterion objective function is a weighted sum of: an inundation coverage objective function that prioritizes sensor locations in geographical cells that yield a highest aggregated normalized inundation score across different combinations of the plurality of geographical cells, and a cluster coverage objective function that prioritizes sensor locations in geographical cells that maximize coverage across the one or more clusters of interconnected geographical cells.

In some embodiments of the computer-program product, the inundation coverage objective function prioritizes the sensor locations in the geographical cells that: are assigned a criticality multiplier value due a presence of a pre-defined type of asset and yield the highest aggregated normalized inundation score across different combinations of the plurality of geographical cells.

In some embodiments of the computer-program product, the optimization algorithm includes: a constraint that constrains the set of optimal sensor locations to a pre-defined number of optimal sensor locations, and a constraint that constrains the set of optimal sensor locations to include one or more pre-determined critical locations.

In some embodiments of the computer-program product, creating the geospatial dataset includes: obtaining a representation of the target spatial area, dividing the representation of the target spatial area into the plurality of geographical cells, obtaining a plurality of feature maps associated with the target spatial area, extracting the features of the plurality of geographical cells from the plurality of feature maps, and adding entries to the geospatial dataset that correspond to the plurality of geographical cells and that store the features extracted for the plurality of geographical cells.

In some embodiments of the computer-program product, the features for the plurality of geographical cells includes features for a respective geographical cell of the plurality of geographical cells, and the features for the respective geographical cell include: one or more identification features associated with the respective geographical cell, one or more topographical features associated with the respective geographical cell, and one or more soil and drainage features associated with the respective geographical cell.

In some embodiments of the computer-program product, the one or more identification features associated with the respective geographical cell include an identification value associated with the respective geographical cell.

In some embodiments of the computer-program product, the one or more topographical features associated with the respective geographical cell include: an elevation of the respective geographical cell, a percentage of impervious surface within the respective geographical cell, and a land cover classification associated with the respective geographical cell.

In some embodiments of the computer-program product, the one or more soil and drainage features associated with the respective geographical cell include: a soil type associated with the respective geographical cell, a type of storm drain present in the respective geographical cell, and a maximum capacity of the storm drain in the respective geographical cell.

In some embodiments of the computer-program product, generating the sensor location map includes overlaying the set of geospatial markers on the plurality of geographical cells; and generating the visual output comprises displaying the sensor location map in a graphical user interface after generating the sensor location map.

In some embodiments of the computer-program product, generating the visual output comprises generating a report comprising a visual depiction of the sensor location map after generating the sensor location map.

In some embodiments of the computer-program product, the computer instructions, when executed by one or more processors, perform additional operations comprising: positioning one or more sensors at one or more of the optimal sensor locations based at least in part on the visual output that displays the sensor location map.

In some embodiments, a computer-implemented method may comprise: creating a geospatial dataset comprising features for a plurality of geographical cells within a target spatial area; executing, via a flood simulation model, a computer simulation that simulates flooding within the target spatial area based on an input of (1) the geospatial dataset and (2) rainfall intensity data of one or more flooding events associated with the target spatial area, wherein executing the computer simulation includes: simulating inundation levels of the plurality of geographical cells during the one or more flooding events, and simulating a flow of water across the plurality of geographical cells during the one or more flooding events; converting the inundation levels of the plurality of geographical cells to normalized inundation scores for the plurality of geographical cells; detecting one or more clusters of interconnected geographical cells based on the simulating of the flow of water; determining, via an optimization algorithm, a set of optimal sensor locations in the target spatial area based on the normalized inundation scores and the one or more clusters of interconnected geographical cells; generating a sensor location map that includes the plurality of geographical cells and a set of geospatial markers identifying the set of optimal sensor locations; and generating a visual output that displays the sensor location map after generating the sensor location map.

In some embodiments of the computer-implemented method, simulating the inundation levels of the plurality of geographical cells during a respective flooding event includes: identifying, from the rainfall intensity data, precipitation amounts observed during a plurality of timesteps of the respective flooding event, simulating, via the flood simulation model, the precipitation amounts observed during the plurality of timesteps on the plurality of geographical cells, and determining maximum inundation levels of the plurality of geographical cells based at least on the simulating of the precipitation amounts.

In some embodiments of the computer-implemented method, determining a maximum inundation level of a respective geographical cell includes: identifying an inundation level of the respective geographical cell at a first timestep of the respective flooding event, identifying inundation levels of the respective geographical cell at a plurality of other timesteps of the respective flooding event, and using the inundation level identified at the first timestep as the maximum inundation level if the inundation level identified at the first timestep is greater than the inundation levels identified at the plurality of other timesteps.

In some embodiments of the computer-implemented method, the inundation levels of the plurality of geographical cells include maximum inundation levels of the plurality of geographical cells during a respective flooding event, and converting an inundation level of a respective geographical cell to a normalized inundation score for the respective geographical cell includes: identifying an overall maximum inundation level among the maximum inundation levels, and computing the normalized inundation score by dividing the

inundation level of the respective geographical cell by the overall maximum inundation level.

In some embodiments of the computer-implemented method, simulating an inundation level of a respective geographical cell includes: identifying precipitation amounts observed during a plurality of timesteps of a respective flooding event, simulating the precipitation amounts observed during the plurality of timesteps on the respective geographical cell, computing inundation levels of the respective geographical cell across the plurality of timesteps, and using a maximum inundation level of the inundation levels computed across the plurality of timesteps as the inundation level of the respective geographical cell.

In some embodiments of the computer-implemented method, computing an inundation level of a respective geographical cell at a respective timestep includes: (a) identifying a precipitation amount observed during the respective timestep, (b) identifying an amount of water flowing from other geographical cells into the respective geographical cell during the respective timestep, (c) identifying an amount of water flowing from the respective geographical cell to the other geographical cells during the respective timestep, (d) identifying an amount of water lost due to infiltration during the respective timestep, (e) identifying an amount of water removed via storm drain outflow during the respective timestep, and (f) computing the inundation level of the respective geographical cell using (a)-(e).

In some embodiments, a computer-implemented system may comprise: one or more processors; a memory; a computer-readable medium operably coupled to the one or more processors, the computer-readable medium having computer-readable instructions stored thereon that, when executed by the one or more processors, cause a computing device to perform operations comprising: creating a geospatial dataset comprising features for a plurality of geographical cells within a target spatial area; executing, via a flood simulation model, a computer simulation that simulates flooding within the target spatial area based on an input of (1) the geospatial dataset and (2) rainfall intensity data of one or more flooding events associated with the target spatial area, wherein executing the computer simulation includes: simulating inundation levels of the plurality of geographical cells during the one or more flooding events, and simulating a flow of water across the plurality of geographical cells during the one or more flooding events; converting the inundation levels of the plurality of geographical cells to normalized inundation scores for the plurality of geographical cells; detecting one or more clusters of interconnected geographical cells based on the simulating of the flow of water; determining, via an optimization algorithm, a set of optimal sensor locations in the target spatial area based on the normalized inundation scores and the one or more clusters of interconnected geographical cells; generating a sensor location map that includes the plurality of geographical cells and a set of geospatial markers identifying the set of optimal sensor locations; and generating a visual output that displays the sensor location map after generating the sensor location map.

BRIEF DESCRIPTION OF THE FIGURES

FIG. 1 illustrates a block diagram that provides an illustration of the hardware components of a computing system, according to some embodiments of the present technology.

FIG. 2 illustrates an example network including an example set of devices communicating with each other over

an exchange system and via a network, according to some embodiments of the present technology.

FIG. 3 illustrates a representation of a conceptual model of a communications protocol system, according to some embodiments of the present technology.

FIG. 4 illustrates a communications grid computing system including a variety of control and worker nodes, according to some embodiments of the present technology.

FIG. 5 illustrates a flow chart showing an example process for adjusting a communications grid or a work project in a communications grid after a failure of a node, according to some embodiments of the present technology.

FIG. 6 illustrates a portion of a communications grid computing system including a control node and a worker node, according to some embodiments of the present technology.

FIG. 7 illustrates a flow chart showing an example process for executing a data analysis or processing project, according to some embodiments of the present technology.

FIG. 8 illustrates a block diagram including components of an Event Stream Processing Engine (ESPE), according to some embodiments of the present technology.

FIG. 9 illustrates a flow chart showing an example process including operations performed by an event stream processing engine, according to some embodiments of the present technology.

FIG. 10 illustrates an ESP system interfacing between a publishing device and multiple event subscribing devices, according to some embodiments of the present technology.

FIG. 11 illustrates a flow chart of an example of a process for generating and using a machine-learning model according to some aspects, according to some embodiments of the present technology.

FIG. 12 illustrates an example of a machine-learning model as a neural network, according to some embodiments of the present technology.

FIG. 13 illustrates various aspects of the use of containers as a mechanism to allocate processing, storage and/or other resources of a processing system to the performance of various analyses, according to some embodiments of the present technology.

FIG. 14 illustrates an example of a method for allocation of flood sensors via distributed parameter flood modeling, according to some embodiments of the present technology.

FIG. 15A illustrates an example of a computer simulation setup process that enables allocation of flood sensors via distributed parameter flood modeling, according to some embodiments of the present technology.

FIG. 15B illustrates an example of a cell clustering process that enables allocation of flood sensors via distributed parameter flood modeling, according to some embodiments of the present technology.

FIG. 15C illustrates an example of a cell cluster map that enables allocation of flood sensors via distributed parameter flood modeling, according to some embodiments of the present technology.

FIGS. 15D-1, 15D-2, and 15E illustrate an example of an inundation normalization process that enables allocation of flood sensors via distributed parameter flood modeling, according to some embodiments of the present technology.

FIG. 15F illustrates an example of a flood sensor placement process that enables allocation of flood sensors via distributed parameter flood modeling, according to some embodiments of the present technology.

FIGS. 15G-1, 15G-2, and 15G-3 illustrate examples of geographical data that enables allocation of flood sensors via

distributed parameter flood modeling, according to some embodiments of the present technology.

FIG. 15H illustrates an example of rainfall intensity data that enables allocation of flood sensors via distributed parameter flood modeling, according to some embodiments of the present technology.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

The following description of the preferred embodiments of the inventions are not intended to limit the inventions to these preferred embodiments, but rather to enable any person skilled in the art to make and use these inventions.

Example Systems

Systems depicted in some of the figures may be provided in various configurations. In some embodiments, the systems may be configured as a distributed system where one or more components of the system are distributed across one or more networks in a cloud computing system.

FIG. 1 is a block diagram that provides an illustration of the hardware components of a data transmission network 100, according to embodiments of the present technology. Data transmission network 100 is a specialized computer system that may be used for processing large amounts of data where a large number of computer processing cycles are required.

Data transmission network 100 may also include computing environment 114. Computing environment 114 may be a specialized computer or other machine that processes the data received within the data transmission network 100. Data transmission network 100 also includes one or more network devices 102. Network devices 102 may include client devices that attempt to communicate with computing environment 114. For example, network devices 102 may send data to the computing environment 114 to be processed, may send signals to the computing environment 114 to control different aspects of the computing environment or the data it is processing, among other reasons. Network devices 102 may interact with the computing environment 114 through a number of ways, such as, for example, over one or more networks 108. As shown in FIG. 1, computing environment 114 may include one or more other systems. For example, computing environment 114 may include a database system 118 and/or a communications grid 120.

In other embodiments, network devices may provide a large amount of data, either all at once or streaming over a period of time (e.g., using event stream processing (ESP), described further with respect to FIGS. 8-10), to the computing environment 114 via networks 108. For example, network devices 102 may include network computers, sensors, databases, or other devices that may transmit or otherwise provide data to computing environment 114. For example, network devices may include local area network devices, such as routers, hubs, switches, or other computer networking devices. These devices may provide a variety of stored or generated data, such as network data or data specific to the network devices themselves. Network devices may also include sensors that monitor their environment or other devices to collect data regarding that environment or those devices, and such network devices may provide data they collect over time. Network devices may also include devices within the internet of things, such as devices within a home automation network. Some of these devices may be referred to as edge devices and may involve edge computing

circuitry. Data may be transmitted by network devices directly to computing environment 114 or to network-attached data stores, such as network-attached data stores 110 for storage so that the data may be retrieved later by the computing environment 114 or other portions of data transmission network 100.

Data transmission network 100 may also include one or more network-attached data stores 110. Network-attached data stores 110 are used to store data to be processed by the computing environment 114 as well as any intermediate or final data generated by the computing system in non-volatile memory. However, in certain embodiments, the configuration of the computing environment 114 allows its operations to be performed such that intermediate and final data results can be stored solely in volatile memory (e.g., RAM), without a requirement that intermediate or final data results be stored to non-volatile types of memory (e.g., disk). This can be useful in certain situations, such as when the computing environment 114 receives ad hoc queries from a user and when responses, which are generated by processing large amounts of data, need to be generated on-the-fly. In this non-limiting situation, the computing environment 114 may be configured to retain the processed information within memory so that responses can be generated for the user at different levels of detail as well as allow a user to interactively query against this information.

Network-attached data stores may store a variety of different types of data organized in a variety of different ways and from a variety of different sources. For example, network-attached data storage may include storage other than primary storage located within computing environment 114 that is directly accessible by processors located therein. Network-attached data storage may include secondary, tertiary or auxiliary storage, such as large hard drives, servers, virtual memory, among other types. Storage devices may include portable or non-portable storage devices, optical storage devices, and various other mediums capable of storing, containing data. A machine-readable storage medium or computer-readable storage medium may include a non-transitory medium in which data can be stored and that does not include carrier waves and/or transitory electronic signals. Examples of a non-transitory medium may include, for example, a magnetic disk or tape, optical storage media such as compact disk or digital versatile disk, flash memory, memory or memory devices. A computer-program product may include code and/or machine-executable instructions that may represent a procedure, a function, a subroutine, a program, a routine, a module, a software package, a class, or any combination of instructions, data structures, or program statements. A code segment may be coupled to another code segment or a hardware circuit by passing and/or receiving information, data, arguments, parameters, or memory contents. Information, arguments, parameters, data, etc. may be passed, forwarded, or transmitted via any suitable means including memory sharing, message passing, token passing, network transmission, among others. Furthermore, the data stores may hold a variety of different types of data. For example, network-attached data stores 110 may hold unstructured (e.g., raw) data, such as manufacturing data (e.g., a database containing records identifying products being manufactured with parameter data for each product, such as colors and models) or product sales databases (e.g., a database containing individual data records identifying details of individual product sales).

The unstructured data may be presented to the computing environment 114 in different forms such as a flat file or a

conglomerate of data records and may have data values and accompanying time stamps. The computing environment 114 may be used to analyze the unstructured data in a variety of ways to determine the best way to structure (e.g., hierarchically) that data, such that the structured data is tailored to a type of further analysis that a user wishes to perform on the data. For example, after being processed, the unstructured time stamped data may be aggregated by time (e.g., into daily time period units) to generate time series data and/or structured hierarchically according to one or more dimensions (e.g., parameters, attributes, and/or variables). For example, data may be stored in a hierarchical data structure, such as a ROLAP OR MOLAP database, or may be stored in another tabular form, such as in a flat-hierarchy form.

Data transmission network 100 may also include one or more server farms 106. Computing environment 114 may route select communications or data to the one or more server farms 106 or one or more servers within the server farms. Server farms 106 can be configured to provide information in a predetermined manner. For example, server farms 106 may access data to transmit in response to a communication. Server farms 106 may be separately housed from each other device within data transmission network 100, such as computing environment 114, and/or may be part of a device or system.

Server farms 106 may host a variety of different types of data processing as part of data transmission network 100. Server farms 106 may receive a variety of different data from network devices, from computing environment 114, from cloud network 116, or from other sources. The data may have been obtained or collected from one or more sensors, as inputs from a control database, or may have been received as inputs from an external system or device. Server farms 106 may assist in processing the data by turning raw data into processed data based on one or more rules implemented by the server farms. For example, sensor data may be analyzed to determine changes in an environment over time or in real-time.

Data transmission network 100 may also include one or more cloud networks 116. Cloud network 116 may include a cloud infrastructure system that provides cloud services. In certain embodiments, services provided by the cloud network 116 may include a host of services that are made available to users of the cloud infrastructure system on demand. Cloud network 116 is shown in FIG. 1 as being connected to computing environment 114 (and therefore having computing environment 114 as its client or user), but cloud network 116 may be connected to or utilized by any of the devices in FIG. 1. Services provided by the cloud network can dynamically scale to meet the needs of its users. The cloud network 116 may include one or more computers, servers, and/or systems. In some embodiments, the computers, servers, and/or systems that make up the cloud network 116 are different from the user's own on-premises computers, servers, and/or systems. For example, the cloud network 116 may host an application, and a user may, via a communication network such as the Internet, on demand, order and use the application.

While each device, server and system in FIG. 1 is shown as a single device, it will be appreciated that multiple devices may instead be used. For example, a set of network devices can be used to transmit various communications from a single user, or remote server 140 may include a server stack. As another example, data may be processed as part of computing environment 114.

11

Each communication within data transmission network 100 (e.g., between client devices, between servers 106 and computing environment 114 or between a server and a device) may occur over one or more networks 108. Networks 108 may include one or more of a variety of different types of networks, including a wireless network, a wired network, or a combination of a wired and wireless network. Examples of suitable networks include the Internet, a personal area network, a local area network (LAN), a wide area network (WAN), or a wireless local area network (WLAN). A wireless network may include a wireless interface or combination of wireless interfaces. As an example, a network in the one or more networks 108 may include a short-range communication channel, such as a BLUETOOTH® communication channel or a BLUETOOTH® LOW Energy communication channel. A wired network may include a wired interface. The wired and/or wireless networks may be implemented using routers, access points, bridges, gateways, or the like, to connect devices in the network 114, as will be further described with respect to FIG. 2. The one or more networks 108 can be incorporated entirely within or can include an intranet, an extranet, or a combination thereof. In one embodiment, communications between two or more systems and/or devices can be achieved by a secure communications protocol, such as secure sockets layer (SSL) or transport layer security (TLS). In addition, data and/or transactional details may be encrypted.

Some aspects may utilize the Internet of Things (IoT), where things (e.g., machines, devices, phones, sensors) can be connected to networks and the data from these things can be collected and processed within the things and/or external to the things. For example, the IoT can include sensors in many different devices, and high value analytics can be applied to identify hidden relationships and drive increased efficiencies. This can apply to both big data analytics and real-time (e.g., ESP) analytics. This will be described further below with respect to FIG. 2.

As noted, computing environment 114 may include a communications grid 120 and a transmission network database system 118. Communications grid 120 may be a grid-based computing system for processing large amounts of data. The transmission network database system 118 may be for managing, storing, and retrieving large amounts of data that are distributed to and stored in the one or more network-attached data stores 110 or other data stores that reside at different locations within the transmission network database system 118. The compute nodes in the grid-based computing system 120 and the transmission network database system 118 may share the same processor hardware, such as processors that are located within computing environment 114.

FIG. 2 illustrates an example network including an example set of devices communicating with each other over an exchange system and via a network, according to embodiments of the present technology. As noted, each communication within data transmission network 100 may occur over one or more networks. System 200 includes a network device 204 configured to communicate with a variety of types of client devices, for example client devices 230, over a variety of types of communication channels.

As shown in FIG. 2, network device 204 can transmit a communication over a network (e.g., a cellular network via a base station 210). The communication can be routed to another network device, such as network devices 205-209, via base station 210. The communication can also be routed to computing environment 214 via base station 210. For

12

example, network device 204 may collect data either from its surrounding environment or from other network devices (such as network devices 205-209) and transmit that data to computing environment 214.

Although network devices 204-209 are shown in FIG. 2 as a mobile phone, laptop computer, tablet computer, temperature sensor, motion sensor, and audio sensor respectively, the network devices may be or include sensors that are sensitive to detecting aspects of their environment. For example, the network devices may include sensors such as water sensors, power sensors, electrical current sensors, chemical sensors, optical sensors, pressure sensors, geographic or position sensors (e.g., GPS), velocity sensors, acceleration sensors, flow rate sensors, among others. Examples of characteristics that may be sensed include force, torque, load, strain, position, temperature, air pressure, fluid flow, chemical properties, resistance, electromagnetic fields, radiation, irradiance, proximity, acoustics, moisture, distance, speed, vibrations, acceleration, electrical potential, and electrical current, among others. The sensors may be mounted to various components used as part of a variety of different types of systems (e.g., an oil drilling operation). The network devices may detect and record data related to the environment that it monitors and transmit that data to computing environment 214.

As noted, one type of system that may include various sensors that collect data to be processed and/or transmitted to a computing environment according to certain embodiments includes an oil drilling system. For example, the one or more drilling operation sensors may include surface sensors that measure a hook load, a fluid rate, a temperature and a density in and out of the wellbore, a standpipe pressure, a surface torque, a rotation speed of a drill pipe, a rate of penetration, a mechanical specific energy, etc. and downhole sensors that measure a rotation speed of a bit, fluid densities, downhole torque, downhole vibration (axial, tangential, lateral), a weight applied at a drill bit, an annular pressure, a differential pressure, an azimuth, an inclination, a dog leg severity, a measured depth, a vertical depth, a downhole temperature, etc. Besides the raw data collected directly by the sensors, other data may include parameters either developed by the sensors or assigned to the system by a client or other controlling device. For example, one or more drilling operation control parameters may control settings such as a mud motor speed to flow ratio, a bit diameter, a predicted formation top, seismic data, weather data, etc. Other data may be generated using physical models such as an earth model, a weather model, a seismic model, a bottom hole assembly model, a well plan model, an annular friction model, etc. In addition to sensor and control settings, predicted outputs, of for example, the rate of penetration, mechanical specific energy, hook load, flow in fluid rate, flow out fluid rate, pump pressure, surface torque, rotation speed of the drill pipe, annular pressure, annular friction pressure, annular temperature, equivalent circulating density, etc. may also be stored in the data warehouse.

In another example, another type of system that may include various sensors that collect data to be processed and/or transmitted to a computing environment according to certain embodiments includes a home automation or similar automated network in a different environment, such as an office space, school, public space, sports venue, or a variety of other locations. Network devices in such an automated network may include network devices that allow a user to access, control, and/or configure various home appliances located within the user's home (e.g., a television, radio, light, fan, humidifier, sensor, microwave, iron, and/or the

13

like), or outside of the user's home (e.g., exterior motion sensors, exterior lighting, garage door openers, sprinkler systems, or the like). For example, network device 102 may include a home automation switch that may be coupled with a home appliance. In another embodiment, a network device can allow a user to access, control, and/or configure devices, such as office-related devices (e.g., copy machine, printer, or fax machine), audio and/or video related devices (e.g., a receiver, a speaker, a projector, a DVD player, or a television), media-playback devices (e.g., a compact disc player, a CD player, or the like), computing devices (e.g., a home computer, a laptop computer, a tablet, a personal digital assistant (PDA), a computing device, or a wearable device), lighting devices (e.g., a lamp or recessed lighting), devices associated with a security system, devices associated with an alarm system, devices that can be operated in an automobile (e.g., radio devices, navigation devices), and/or the like. Data may be collected from such various sensors in raw form, or data may be processed by the sensors to create parameters or other data either developed by the sensors based on the raw data or assigned to the system by a client or other controlling device.

In another example, another type of system that may include various sensors that collect data to be processed and/or transmitted to a computing environment according to certain embodiments includes a power or energy grid. A variety of different network devices may be included in an energy grid, such as various devices within one or more power plants, energy farms (e.g., wind farm, solar farm, among others) energy storage facilities, factories, homes and businesses of consumers, among others. One or more of such devices may include one or more sensors that detect energy gain or loss, electrical input or output or loss, and a variety of other efficiencies. These sensors may collect data to inform users of how the energy grid, and individual devices within the grid, may be functioning and how they may be made more efficient.

Network device sensors may also perform processing on data it collects before transmitting the data to the computing environment 114, or before deciding whether to transmit data to the computing environment 114. For example, network devices may determine whether data collected meets certain rules, for example by comparing data or values calculated from the data and comparing that data to one or more thresholds. The network device may use this data and/or comparisons to determine if the data should be transmitted to the computing environment 214 for further use or processing.

Computing environment 214 may include machines 220 and 240. Although computing environment 214 is shown in FIG. 2 as having two machines, 220 and 240, computing environment 214 may have only one machine or may have more than two machines. The machines that make up computing environment 214 may include specialized computers, servers, or other machines that are configured to individually and/or collectively process large amounts of data. The computing environment 214 may also include storage devices that include one or more databases of structured data, such as data organized in one or more hierarchies, or unstructured data. The databases may communicate with the processing devices within computing environment 214 to distribute data to them. Since network devices may transmit data to computing environment 214, that data may be received by the computing environment 214 and subsequently stored within those storage devices. Data used by computing environment 214 may also be

14

stored in data stores 235, which may also be a part of or connected to computing environment 214.

Computing environment 214 can communicate with various devices via one or more routers 225 or other internetwork or intra-network connection components. For example, computing environment 214 may communicate with devices 230 via one or more routers 225. Computing environment 214 may collect, analyze and/or store data from or pertaining to communications, client device operations, client rules, and/or user-associated actions stored at one or more data stores 235. Such data may influence communication routing to the devices within computing environment 214, how data is stored or processed within computing environment 214, among other actions.

Notably, various other devices can further be used to influence communication routing and/or processing between devices within computing environment 214 and with devices outside of computing environment 214. For example, as shown in FIG. 2, computing environment 214 may include a web server 240. Thus, computing environment 214 can retrieve data of interest, such as client information (e.g., product information, client rules, etc.), technical product details, news, current or predicted weather, and so on.

In addition to computing environment 214 collecting data (e.g., as received from network devices, such as sensors, and client devices or other sources) to be processed as part of a big data analytics project, it may also receive data in real time as part of a streaming analytics environment. As noted, data may be collected using a variety of sources as communicated via different kinds of networks or locally. Such data may be received on a real-time streaming basis. For example, network devices may receive data periodically from network device sensors as the sensors continuously sense, monitor and track changes in their environments. Devices within computing environment 214 may also perform pre-analysis on data it receives to determine if the data received should be processed as part of an ongoing project. The data received and collected by computing environment 214, no matter what the source or method or timing of receipt, may be processed over a period of time for a client to determine results data based on the client's needs and rules.

FIG. 3 illustrates a representation of a conceptual model of a communications protocol system, according to embodiments of the present technology. More specifically, FIG. 3 identifies operation of a computing environment in an Open Systems Interaction model that corresponds to various connection components. The model 300 shows, for example, how a computing environment, such as computing environment 314 (or computing environment 214 in FIG. 2) may communicate with other devices in its network, and control how communications between the computing environment and other devices are executed and under what conditions.

The model can include layers 301-307. The layers are arranged in a stack. Each layer in the stack serves the layer one level higher than it (except for the application layer, which is the highest layer), and is served by the layer one level below it (except for the physical layer, which is the lowest layer). The physical layer is the lowest layer because it receives and transmits raw bites of data and is the farthest layer from the user in a communications system. On the other hand, the application layer is the highest layer because it interacts directly with a software application.

As noted, the model includes a physical layer 301. Physical layer 301 represents physical communication and can define parameters of that physical communication. For example, such physical communication may come in the

15

form of electrical, optical, or electromagnetic signals. Physical layer 301 also defines protocols that may control communications within a data transmission network.

Link layer 302 defines links and mechanisms used to transmit (i.e., move) data across a network. The link layer 302 manages node-to-node communications, such as within a grid computing environment. Link layer 302 can detect and correct errors (e.g., transmission errors in the physical layer 301). Link layer 302 can also include a media access control (MAC) layer and logical link control (LLC) layer.

Network layer 303 defines the protocol for routing within a network. In other words, the network layer coordinates transferring data across nodes in a same network (e.g., such as a grid computing environment). Network layer 303 can also define the processes used to structure local addressing within the network.

Transport layer 304 can manage the transmission of data and the quality of the transmission and/or receipt of that data. Transport layer 304 can provide a protocol for transferring data, such as, for example, a Transmission Control Protocol (TCP). Transport layer 304 can assemble and disassemble data frames for transmission. The transport layer can also detect transmission errors occurring in the layers below it.

Session layer 305 can establish, maintain, and manage communication connections between devices on a network. In other words, the session layer controls the dialogues or nature of communications between network devices on the network. The session layer may also establish checkpointing, adjournment, termination, and restart procedures.

Presentation layer 306 can provide translation for communications between the application and network layers. In other words, this layer may encrypt, decrypt and/or format data based on data types and/or encodings known to be accepted by an application or network layer.

Application layer 307 interacts directly with software applications and end users and manages communications between them. Application layer 307 can identify destinations, local resource states or availability and/or communication content or formatting using the applications.

Intra-network connection components 321 and 322 are shown to operate in lower levels, such as physical layer 301 and link layer 302, respectively. For example, a hub can operate in the physical layer, a switch can operate in the link layer, and a router can operate in the network layer. Internetwork connection components 323 and 328 are shown to operate on higher levels, such as layers 303-307. For example, routers can operate in the network layer and network devices can operate in the transport, session, presentation, and application layers.

As noted, a computing environment 314 can interact with and/or operate on, in various embodiments, one, more, all or any of the various layers. For example, computing environment 314 can interact with a hub (e.g., via the link layer) so as to adjust which devices the hub communicates with. The physical layer may be served by the link layer, so it may implement such data from the link layer. For example, the computing environment 314 may control which devices it will receive data from. For example, if the computing environment 314 knows that a certain network device has turned off, broken, or otherwise become unavailable or unreliable, the computing environment 314 may instruct the hub to prevent any data from being transmitted to the computing environment 314 from that network device. Such a process may be beneficial to avoid receiving data that is inaccurate or that has been influenced by an uncontrolled environment. As another example, computing environment

16

314 can communicate with a bridge, switch, router or gateway and influence which device within the system (e.g., system 200) the component selects as a destination. In some embodiments, computing environment 314 can interact with various layers by exchanging communications with equipment operating on a particular layer by routing or modifying existing communications. In another embodiment, such as in a grid computing environment, a node may determine how data within the environment should be routed (e.g., which node should receive certain data) based on certain parameters or information provided by other layers within the model.

As noted, the computing environment 314 may be a part of a communications grid environment, the communications of which may be implemented as shown in the protocol of FIG. 3. For example, referring back to FIG. 2, one or more machines 220 and 240 may be part of a communications grid computing environment. A gridded computing environment may be employed in a distributed system with non-interactive workloads where data resides in memory on the machines, or compute nodes. In such an environment, analytic code, instead of a database management system, controls the processing performed by the nodes. Data is collocated by pre-distributing it to the grid nodes, and the analytic code on each node loads the local data into memory. Each node may be assigned a particular task such as a portion of a processing project, or to organize or control other nodes within the grid.

FIG. 4 illustrates a communications grid computing system 400 including a variety of control and worker nodes, according to embodiments of the present technology. Communications grid computing system 400 includes three control nodes and one or more worker nodes. Communications grid computing system 400 includes control nodes 402, 404, and 406. The control nodes are communicatively connected via communication paths 451, 453, and 455. Therefore, the control nodes may transmit information (e.g., related to the communications grid or notifications), to and receive information from each other. Although communications grid computing system 400 is shown in FIG. 4 as including three control nodes, the communications grid may include more or less than three control nodes.

Communications grid computing system (or just "communications grid") 400 also includes one or more worker nodes. Shown in FIG. 4 are six worker nodes 410-420. Although FIG. 4 shows six worker nodes, a communications grid according to embodiments of the present technology may include more or less than six worker nodes. The number of worker nodes included in a communications grid may be dependent upon how large the project or data set is being processed by the communications grid, the capacity of each worker node, the time designated for the communications grid to complete the project, among others. Each worker node within the communications grid 400 may be connected (wired or wirelessly, and directly or indirectly) to control nodes 402-406. Therefore, each worker node may receive information from the control nodes (e.g., an instruction to perform work on a project) and may transmit information to the control nodes (e.g., a result from work performed on a project). Furthermore, worker nodes may communicate with each other (either directly or indirectly). For example, worker nodes may transmit data between each other related to a job being performed or an individual task within a job being performed by that worker node. However, in certain embodiments, worker nodes may not, for example, be connected (communicatively or otherwise) to certain other worker nodes. In an embodiment, worker nodes may only be

able to communicate with the control node that controls it and may not be able to communicate with other worker nodes in the communications grid, whether they are other worker nodes controlled by the control node that controls the worker node, or worker nodes that are controlled by other control nodes in the communications grid.

A control node may connect with an external device with which the control node may communicate (e.g., a grid user, such as a server or computer, may connect to a controller of the grid). For example, a server or computer may connect to control nodes and may transmit a project or job to the node. The project may include a data set. The data set may be of any size. Once the control node receives such a project including a large data set, the control node may distribute the data set or projects related to the data set to be performed by worker nodes. Alternatively, for a project including a large data set, the data set may be received or stored by a machine other than a control node (e.g., a HADOOP® standard-compliant data node employing the HADOOP® Distributed File System, or HDFS).

Control nodes may maintain knowledge of the status of the nodes in the grid (i.e., grid status information), accept work requests from clients, subdivide the work across worker nodes, and coordinate the worker nodes, among other responsibilities. Worker nodes may accept work requests from a control node and provide the control node with results of the work performed by the worker node. A grid may be started from a single node (e.g., a machine, computer, server, etc.). This first node may be assigned or may start as the primary control node that will control any additional nodes that enter the grid.

When a project is submitted for execution (e.g., by a client or a controller of the grid) it may be assigned to a set of nodes. After the nodes are assigned to a project, a data structure (i.e., a communicator) may be created. The communicator may be used by the project for information to be shared between the project codes running on each node. A communication handle may be created on each node. A handle, for example, is a reference to the communicator that is valid within a single process on a single node, and the handle may be used when requesting communications between nodes.

A control node, such as control node 402, may be designated as the primary control node. A server, computer or other external device may connect to the primary control node. Once the control node receives a project, the primary control node may distribute portions of the project to its worker nodes for execution. For example, when a project is initiated on communications grid 400, primary control node 402 controls the work to be performed for the project in order to complete the project as requested or instructed. The primary control node may distribute work to the worker nodes based on various factors, such as which subsets or portions of projects may be completed most efficiently and in the correct amount of time. For example, a worker node may perform analysis on a portion of data that is already local (e.g., stored on) the worker node. The primary control node also coordinates and processes the results of the work performed by each worker node after each worker node executes and completes its job. For example, the primary control node may receive a result from one or more worker nodes, and the control node may organize (e.g., collect and assemble) the results received and compile them to produce a complete result for the project received from the end user.

Any remaining control nodes, such as control nodes 404 and 406, may be assigned as backup control nodes for the project. In an embodiment, backup control nodes may not

control any portion of the project. Instead, backup control nodes may serve as a backup for the primary control node and take over as primary control node if the primary control node were to fail. If a communications grid were to include only a single control node, and the control node were to fail (e.g., the control node is shut off or breaks) then the communications grid as a whole may fail and any project or job being run on the communications grid may fail and may not complete. While the project may be run again, such a failure may cause a delay (severe delay in some cases, such as overnight delay) in completion of the project. Therefore, a grid with multiple control nodes, including a backup control node, may be beneficial.

To add another node or machine to the grid, the primary control node may open a pair of listening sockets, for example. A socket may be used to accept work requests from clients, and the second socket may be used to accept connections from other grid nodes. The primary control node may be provided with a list of other nodes (e.g., other machines, computers, servers) that will participate in the grid, and the role that each node will fill in the grid. Upon startup of the primary control node (e.g., the first node on the grid), the primary control node may use a network protocol to start the server process on every other node in the grid. Command line parameters, for example, may inform each node of one or more pieces of information, such as: the role that the node will have in the grid, the host name of the primary control node, the port number on which the primary control node is accepting connections from peer nodes, among others. The information may also be provided in a configuration file, transmitted over a secure shell tunnel, recovered from a configuration server, among others. While the other machines in the grid may not initially know about the configuration of the grid, that information may also be sent to each other node by the primary control node. Updates of the grid information may also be subsequently sent to those nodes.

For any control node other than the primary control node added to the grid, the control node may open three sockets. The first socket may accept work requests from clients, the second socket may accept connections from other grid members, and the third socket may connect (e.g., permanently) to the primary control node. When a control node (e.g., primary control node) receives a connection from another control node, it first checks to see if the peer node is in the list of configured nodes in the grid. If it is not on the list, the control node may clear the connection. If it is on the list, it may then attempt to authenticate the connection. If authentication is successful, the authenticating node may transmit information to its peer, such as the port number on which a node is listening for connections, the host name of the node, information about how to authenticate the node, among other information. When a node, such as the new control node, receives information about another active node, it will check to see if it already has a connection to that other node. If it does not have a connection to that node, it may then establish a connection to that control node.

Any worker node added to the grid may establish a connection to the primary control node and any other control nodes on the grid. After establishing the connection, it may authenticate itself to the grid (e.g., any control nodes, including both primary and backup, or a server or user controlling the grid). After successful authentication, the worker node may accept configuration information from the control node.

When a node joins a communications grid (e.g., when the node is powered on or connected to an existing node on the

grid or both), the node is assigned (e.g., by an operating system of the grid) a universally unique identifier (UUID). This unique identifier may help other nodes and external entities (devices, users, etc.) to identify the node and distinguish it from other nodes. When a node is connected to the grid, the node may share its unique identifier with the other nodes in the grid. Since each node may share its unique identifier, each node may know the unique identifier of every other node on the grid. Unique identifiers may also designate a hierarchy of each of the nodes (e.g., backup control nodes) within the grid. For example, the unique identifiers of each of the backup control nodes may be stored in a list of backup control nodes to indicate an order in which the backup control nodes will take over for a failed primary control node to become a new primary control node. However, a hierarchy of nodes may also be determined using methods other than using the unique identifiers of the nodes. For example, the hierarchy may be predetermined or may be assigned based on other predetermined factors.

The grid may add new machines at any time (e.g., initiated from any control node). Upon adding a new node to the grid, the control node may first add the new node to its table of grid nodes. The control node may also then notify every other control node about the new node. The nodes receiving the notification may acknowledge that they have updated their configuration information.

Primary control node 402 may, for example, transmit one or more communications to backup control nodes 404 and 406 (and, for example, to other control or worker nodes within the communications grid). Such communications may be sent periodically, at fixed time intervals, between known fixed stages of the project's execution, among other protocols. The communications transmitted by primary control node 402 may be of varied types and may include a variety of types of information. For example, primary control node 402 may transmit snapshots (e.g., status information) of the communications grid so that backup control node 404 always has a recent snapshot of the communications grid. The snapshot or grid status may include, for example, the structure of the grid (including, for example, the worker nodes in the grid, unique identifiers of the nodes, or their relationships with the primary control node) and the status of a project (including, for example, the status of each worker node's portion of the project). The snapshot may also include analysis or results received from worker nodes in the communications grid. The backup control nodes may receive and store the backup data received from the primary control node. The backup control nodes may transmit a request for such a snapshot (or other information) from the primary control node, or the primary control node may send such information periodically to the backup control nodes.

As noted, the backup data may allow the backup control node to take over as primary control node if the primary control node fails without requiring the grid to start the project over from scratch. If the primary control node fails, the backup control node that will take over as primary control node may retrieve the most recent version of the snapshot received from the primary control node and use the snapshot to continue the project from the stage of the project indicated by the backup data. This may prevent failure of the project as a whole.

A backup control node may use various methods to determine that the primary control node has failed. In one example of such a method, the primary control node may transmit (e.g., periodically) a communication to the backup control node that indicates that the primary control node is working and has not failed, such as a heartbeat communica-

cation. The backup control node may determine that the primary control node has failed if the backup control node has not received a heartbeat communication for a certain predetermined period of time. Alternatively, a backup control node may also receive a communication from the primary control node itself (before it failed) or from a worker node that the primary control node has failed, for example because the primary control node has failed to communicate with the worker node.

Different methods may be performed to determine which backup control node of a set of backup control nodes (e.g., backup control nodes 404 and 406) will take over for failed primary control node 402 and become the new primary control node. For example, the new primary control node 15 may be chosen based on a ranking or "hierarchy" of backup control nodes based on their unique identifiers. In an alternative embodiment, a backup control node may be assigned to be the new primary control node by another device in the communications grid or from an external device (e.g., a 20 system infrastructure or an end user, such as a server or computer, controlling the communications grid). In another alternative embodiment, the backup control node that takes over as the new primary control node may be designated based on bandwidth or other statistics about the communications grid.

A worker node within the communications grid may also fail. If a worker node fails, work being performed by the failed worker node may be redistributed amongst the operational worker nodes. In an alternative embodiment, the 30 primary control node may transmit a communication to each of the operable worker nodes still on the communications grid that each of the worker nodes should purposefully fail also. After each of the worker nodes fail, they may each retrieve their most recently saved checkpoint of their status and re-start the project from that checkpoint to minimize lost 35 progress on the project being executed.

FIG. 5 illustrates a flow chart showing an example process 500 for adjusting a communications grid or a work project in a communications grid after a failure of a node, according 40 to embodiments of the present technology. The process may include, for example, receiving grid status information including a project status of a portion of a project being executed by a node in the communications grid, as described in operation 502. For example, a control node (e.g., a backup 45 control node connected to a primary control node and a worker node on a communications grid) may receive grid status information, where the grid status information includes a project status of the primary control node or a project status of the worker node. The project status of the primary control node and the project status of the worker node may include a status of one or more portions of a 50 project being executed by the primary and worker nodes in the communications grid. The process may also include storing the grid status information, as described in operation 504. For example, a control node (e.g., a backup control node) may store the received grid status information locally 55 within the control node. Alternatively, the grid status information may be sent to another device for storage where the control node may have access to the information.

The process may also include receiving a failure communication corresponding to a node in the communications grid in operation 506. For example, a node may receive a failure communication including an indication that the primary control node has failed, prompting a backup control node to take over for the primary control node. In an alternative embodiment, a node may receive a failure that a worker 60 node has failed, prompting a control node to reassign the

work being performed by the worker node. The process may also include reassigning a node or a portion of the project being executed by the failed node, as described in operation 508. For example, a control node may designate the backup control node as a new primary control node based on the failure communication upon receiving the failure communication. If the failed node is a worker node, a control node may identify a project status of the failed worker node using the snapshot of the communications grid, where the project status of the failed worker node includes a status of a portion of the project being executed by the failed worker node at the failure time.

The process may also include receiving updated grid status information based on the reassignment, as described in operation 510, and transmitting a set of instructions based on the updated grid status information to one or more nodes in the communications grid, as described in operation 512. The updated grid status information may include an updated project status of the primary control node or an updated project status of the worker node. The updated information may be transmitted to the other nodes in the grid to update their stale stored information.

FIG. 6 illustrates a portion of a communications grid computing system 600 including a control node and a worker node, according to embodiments of the present technology. Communications grid 600 computing system includes one control node (control node 602) and one worker node (worker node 610) for purposes of illustration but may include more worker and/or control nodes. The control node 602 is communicatively connected to worker node 610 via communication path 650. Therefore, control node 602 may transmit information (e.g., related to the communications grid or notifications), to and receive information from worker node 610 via path 650.

Similar to in FIG. 4, communications grid computing system (or just “communications grid”) 600 includes data processing nodes (control node 602 and worker node 610). Nodes 602 and 610 include multi-core data processors. Each node 602 and 610 includes a grid-enabled software component (GESC) 620 that executes on the data processor associated with that node and interfaces with buffer memory 622 also associated with that node. Each node 602 and 610 includes database management software (DBMS) 628 that executes on a database server (not shown) at control node 602 and on a database server (not shown) at worker node 610.

Each node also includes a data store 624. Data stores 624, similar to network-attached data stores 110 in FIG. 1 and data stores 235 in FIG. 2, are used to store data to be processed by the nodes in the computing environment. Data stores 624 may also store any intermediate or final data generated by the computing system after being processed, for example in non-volatile memory. However, in certain embodiments, the configuration of the grid computing environment allows its operations to be performed such that intermediate and final data results can be stored solely in volatile memory (e.g., RAM), without a requirement that intermediate or final data results be stored to non-volatile types of memory. Storing such data in volatile memory may be useful in certain situations, such as when the grid receives queries (e.g., ad hoc) from a client and when responses, which are generated by processing large amounts of data, need to be generated quickly or on-the-fly. In such a situation, the grid may be configured to retain the data within memory so that responses can be generated at different levels of detail and so that a client may interactively query against this information.

Each node also includes a user-defined function (UDF) 626. The UDF provides a mechanism for the DBMS 628 to transfer data to or receive data from the database stored in the data stores 624 that are managed by the DBMS. For example, UDF 626 can be invoked by the DBMS to provide data to the GESC for processing. The UDF 626 may establish a socket connection (not shown) with the GESC to transfer the data. Alternatively, the UDF 626 can transfer data to the GESC by writing data to shared memory accessible by both the UDF and the GESC.

The GESC 620 at the nodes 602 and 620 may be connected via a network, such as network 108 shown in FIG. 1. Therefore, nodes 602 and 620 can communicate with each other via the network using a predetermined communication protocol such as, for example, the Message Passing Interface (MPI). Each GESC 620 can engage in point-to-point communication with the GESC at another node or in collective communication with multiple GESCs via the network. The GESC 620 at each node may contain identical (or nearly identical) software instructions. Each node may be capable of operating as either a control node or a worker node. The GESC at the control node 602 can communicate, over a communication path 652, with a client device 630. More specifically, control node 602 may communicate with client application 632 hosted by the client device 630 to receive queries and to respond to those queries after processing large amounts of data.

DBMS 628 may control the creation, maintenance, and use of database or data structure (not shown) within a node 602 or 610. The database may organize data stored in data stores 624. The DBMS 628 at control node 602 may accept requests for data and transfer the appropriate data for the request. With such a process, collections of data may be distributed across multiple physical locations. In this example, each node 602 and 610 stores a portion of the total data managed by the management system in its associated data store 624.

Furthermore, the DBMS may be responsible for protecting against data loss using replication techniques. Replication includes providing a backup copy of data stored on one node on one or more other nodes. Therefore, if one node fails, the data from the failed node can be recovered from a replicated copy residing at another node. However, as described herein with respect to FIG. 4, data or status information for each node in the communications grid may also be shared with each node on the grid.

FIG. 7 illustrates a flow chart showing an example method 700 for executing a project within a grid computing system, according to embodiments of the present technology. As described with respect to FIG. 6, the GESC at the control node may transmit data with a client device (e.g., client device 630) to receive queries for executing a project and to respond to those queries after large amounts of data have been processed. The query may be transmitted to the control node, where the query may include a request for executing a project, as described in operation 702. The query can contain instructions on the type of data analysis to be performed in the project and whether the project should be executed using the grid-based computing environment, as shown in operation 704.

To initiate the project, the control node may determine if the query requests use of the grid-based computing environment to execute the project. If the determination is no, then the control node initiates execution of the project in a solo environment (e.g., at the control node), as described in operation 710. If the determination is yes, the control node may initiate execution of the project in the grid-based

computing environment, as described in operation 706. In such a situation, the request may include a requested configuration of the grid. For example, the request may include a number of control nodes and a number of worker nodes to be used in the grid when executing the project. After the project has been completed, the control node may transmit results of the analysis yielded by the grid, as described in operation 708. Whether the project is executed in a solo or grid-based environment, the control node provides the results of the project, as described in operation 712.

As noted with respect to FIG. 2, the computing environments described herein may collect data (e.g., as received from network devices, such as sensors, such as network devices 204-209 in FIG. 2, and client devices or other sources) to be processed as part of a data analytics project, and data may be received in real time as part of a streaming analytics environment (e.g., ESP). Data may be collected using a variety of sources as communicated via different kinds of networks or locally, such as on a real-time streaming basis. For example, network devices may receive data periodically from network device sensors as the sensors continuously sense, monitor and track changes in their environments. More specifically, an increasing number of distributed applications develop or produce continuously flowing data from distributed sources by applying queries to the data before distributing the data to geographically distributed recipients. An event stream processing engine (ESPE) may continuously apply the queries to the data as it is received and determines which entities should receive the data. A client or other devices may also subscribe to the ESPE or other devices processing ESP data so that they can receive data after processing, based on for example the entities determined by the processing engine. For example, client devices 230 in FIG. 2 may subscribe to the ESPE in computing environment 214. In another example, event subscription devices 1024a-c, described further with respect to FIG. 10, may also subscribe to the ESPE. The ESPE may determine or define how input data or event streams from network devices or other publishers (e.g., network devices 204-209 in FIG. 2) are transformed into meaningful output data to be consumed by subscribers, such as for example client devices 230 in FIG. 2.

FIG. 8 illustrates a block diagram including components of an Event Stream Processing Engine (ESPE), according to embodiments of the present technology. ESPE 800 may include one or more projects 802. A project may be described as a second-level container in an engine model managed by ESPE 800 where a thread pool size for the project may be defined by a user. Each project of the one or more projects 802 may include one or more continuous queries 804 that contain data flows, which are data transformations of incoming event streams. The one or more continuous queries 804 may include one or more source windows 806 and one or more derived windows 808.

The ESPE may receive streaming data over a period of time related to certain events, such as events or other data sensed by one or more network devices. The ESPE may perform operations associated with processing data created by the one or more devices. For example, the ESPE may receive data from the one or more network devices 204-209 shown in FIG. 2. As noted, the network devices may include sensors that sense different aspects of their environments and may collect data over time based on those sensed observations. For example, the ESPE may be implemented within one or more of machines 220 and 240 shown in FIG. 2. The ESPE may be implemented within such a machine by an ESP application. An ESP application may embed an

ESPE with its own dedicated thread pool or pools into its application space where the main application thread can do application-specific work and the ESPE processes event streams at least by creating an instance of a model into processing objects.

5 The engine container is the top-level container in a model that manages the resources of the one or more projects 802. In an illustrative embodiment, for example, there may be only one ESPE 800 for each instance of the ESP application, and ESPE 800 may have a unique engine name. Additionally, the one or more projects 802 may each have unique project names, and each query may have a unique continuous query name and begin with a uniquely named source window of the one or more source windows 806. ESPE 800 10 may or may not be persistent.

15 Continuous query modeling involves defining directed graphs of windows for event stream manipulation and transformation. A window in the context of event stream manipulation and transformation is a processing node in an event stream processing model. A window in a continuous query can perform aggregations, computations, pattern-matching, and other operations on data flowing through the window. A continuous query may be described as a directed graph of source, relational, pattern matching, and procedural 20 windows. The one or more source windows 806 and the one or more derived windows 808 represent continuously 25 executing queries that generate updates to a query result set as new event blocks stream through ESPE 800. A directed graph, for example, is a set of nodes connected by edges, 30 where the edges have a direction associated with them.

An event object may be described as a packet of data accessible as a collection of fields, with at least one of the fields defined as a key or unique identifier (ID). The event object may be created using a variety of formats including 35 binary, alphanumeric, XML, etc. Each event object may include one or more fields designated as a primary identifier (ID) for the event so ESPE 800 can support operation codes (opcodes) for events including insert, update, upsert, and delete. Upsert opcodes update the event if the key field 40 already exists; otherwise, the event is inserted. For illustration, an event object may be a packed binary representation 45 of a set of field values and include both metadata and field data associated with an event. The metadata may include an opcode indicating if the event represents an insert, update, delete, or upsert, a set of flags indicating if the event is a normal, partial-update, or a retention generated event from retention policy management, and a set of microsecond timestamps that can be used for latency measurements.

An event block object may be described as a grouping or 50 package of event objects. An event stream may be described as a flow of event block objects. A continuous query of the one or more continuous queries 804 transforms a source event stream made up of streaming event block objects published into ESPE 800 into one or more output event streams using the one or more source windows 806 and the one or more derived windows 808. A continuous query can 55 also be thought of as data flow modeling.

The one or more source windows 806 are at the top of the directed graph and have no windows feeding into them. 60 Event streams are published into the one or more source windows 806, and from there, the event streams may be directed to the next set of connected windows as defined by the directed graph. The one or more derived windows 808 are all instantiated windows that are not source windows and 65 that have other windows streaming events into them. The one or more derived windows 808 may perform computations or transformations on the incoming event streams. The

one or more derived windows **808** transform event streams based on the window type (that is operators such as join, filter, compute, aggregate, copy, pattern match, procedural, union, etc.) and window settings. As event streams are published into ESPE **800**, they are continuously queried, and the resulting sets of derived windows in these queries are continuously updated.

FIG. 9 illustrates a flow chart showing an example process including operations performed by an event stream processing engine, according to some embodiments of the present technology. As noted, the ESPE **800** (or an associated ESP application) defines how input event streams are transformed into meaningful output event streams. More specifically, the ESP application may define how input event streams from publishers (e.g., network devices providing sensed data) are transformed into meaningful output event streams consumed by subscribers (e.g., a data analytics project being executed by a machine or set of machines).

Within the application, a user may interact with one or more user interface windows presented to the user in a display under control of the ESPE independently or through a browser application in an order selectable by the user. For example, a user may execute an ESP application, which causes presentation of a first user interface window, which may include a plurality of menus and selectors such as drop-down menus, buttons, text boxes, hyperlinks, etc. associated with the ESP application as understood by a person of skill in the art. As further understood by a person of skill in the art, various operations may be performed in parallel, for example, using a plurality of threads.

At operation **900**, an ESP application may define and start an ESPE, thereby instantiating an ESPE at a device, such as machine **220** and/or **240**. In an operation **902**, the engine container is created. For illustration, ESPE **800** may be instantiated using a function call that specifies the engine container as a manager for the model.

In an operation **904**, the one or more continuous queries **804** are instantiated by ESPE **800** as a model. The one or more continuous queries **804** may be instantiated with a dedicated thread pool or pools that generate updates as new events stream through ESPE **800**. For illustration, the one or more continuous queries **804** may be created to model business processing logic within ESPE **800**, to predict events within ESPE **800**, to model a physical system within ESPE **800**, to predict the physical system state within ESPE **800**, etc. For example, as noted, ESPE **800** may be used to support sensor data monitoring and management (e.g., sensing may include force, torque, load, strain, position, temperature, air pressure, fluid flow, chemical properties, resistance, electromagnetic fields, radiation, irradiance, proximity, acoustics, moisture, distance, speed, vibrations, acceleration, electrical potential, or electrical current, etc.).

ESPE **800** may analyze and process events in motion or “event streams.” Instead of storing data and running queries against the stored data, ESPE **800** may store queries and stream data through them to allow continuous analysis of data as it is received. The one or more source windows **806** and the one or more derived windows **808** may be created based on the relational, pattern matching, and procedural algorithms that transform the input event streams into the output event streams to model, simulate, score, test, predict, etc. based on the continuous query model defined and application to the streamed data.

In an operation **906**, a publish/subscribe (pub/sub) capability is initialized for ESPE **800**. In an illustrative embodiment, a pub/sub capability is initialized for each project of the one or more projects **802**. To initialize and enable

pub/sub capability for ESPE **800**, a port number may be provided. Pub/sub clients can use a host name of an ESP device running the ESPE and the port number to establish pub/sub connections to ESPE **800**.

FIG. 10 illustrates an ESP system **1000** interfacing between publishing device **1022** and event subscribing devices **1024a-c**, according to embodiments of the present technology. ESP system **1000** may include ESP device or subsystem **851**, event publishing device **1022**, an event subscribing device A **1024a**, an event subscribing device B **1024b**, and an event subscribing device C **1024c**. Input event streams are output to ESP device **851** by publishing device **1022**. In alternative embodiments, the input event streams may be created by a plurality of publishing devices. The plurality of publishing devices further may publish event streams to other ESP devices. The one or more continuous queries instantiated by ESPE **800** may analyze and process the input event streams to form output event streams output to event subscribing device A **1024a**, event subscribing device B **1024b**, and event subscribing device C **1024c**. ESP system **1000** may include a greater or a fewer number of event subscribing devices of event subscribing devices.

Publish-subscribe is a message-oriented interaction paradigm based on indirect addressing. Processed data recipients specify their interest in receiving information from ESPE **800** by subscribing to specific classes of events, while information sources publish events to ESPE **800** without directly addressing the receiving parties. ESPE **800** coordinates the interactions and processes the data. In some cases, the data source receives confirmation that the published information has been received by a data recipient.

A publish/subscribe API may be described as a library that enables an event publisher, such as publishing device **1022**, to publish event streams into ESPE **800** or an event subscriber, such as event subscribing device A **1024a**, event subscribing device B **1024b**, and event subscribing device C **1024c**, to subscribe to event streams from ESPE **800**. For illustration, one or more publish/subscribe APIs may be defined. Using the publish/subscribe API, an event publishing application may publish event streams into a running event stream processor project source window of ESPE **800**, and the event subscription application may subscribe to an event stream processor project source window of ESPE **800**.

The publish/subscribe API provides cross-platform connectivity and endianness compatibility between ESP application and other networked applications, such as event publishing applications instantiated at publishing device **1022**, and event subscription applications instantiated at one or more of event subscribing device A **1024a**, event subscribing device B **1024b**, and event subscribing device C **1024c**.

Referring back to FIG. 9, operation **906** initializes the publish/subscribe capability of ESPE **800**. In an operation **908**, the one or more projects **802** are started. The one or more started projects may run in the background on an ESP device. In an operation **910**, an event block object is received from one or more computing devices of the event publishing device **1022**.

ESP subsystem **800** may include a publishing client **1002**, ESPE **800**, a subscribing client A **1004**, a subscribing client B **1006**, and a subscribing client C **1008**. Publishing client **1002** may be started by an event publishing application executing at publishing device **1022** using the publish/subscribe API. Subscribing client A **1004** may be started by an event subscription application A, executing at event subscribing device A **1024a** using the publish/subscribe API. Subscribing client B **1006** may be started by an event

subscription application B executing at event subscribing device B **1024b** using the publish/subscribe API. Subscribing client C **1008** may be started by an event subscription application C executing at event subscribing device C **1024c** using the publish/subscribe API.

An event block object containing one or more event objects is injected into a source window of the one or more source windows **806** from an instance of an event publishing application on event publishing device **1022**. The event block object may be generated, for example, by the event publishing application and may be received by publishing client **1002**. A unique ID may be maintained as the event block object is passed between the one or more source windows **806** and/or the one or more derived windows **808** of ESPE **800**, and to subscribing client A **1004**, subscribing client B **1006**, and subscribing client C **1008** and to event subscription device A **1024a**, event subscription device B **1024b**, and event subscription device C **1024c**. Publishing client **1002** may further generate and include a unique embedded transaction ID in the event block object as the event block object is processed by a continuous query, as well as the unique ID that publishing device **1022** assigned to the event block object.

In an operation **912**, the event block object is processed through the one or more continuous queries **804**. In an operation **914**, the processed event block object is output to one or more computing devices of the event subscribing devices **1024a-c**. For example, subscribing client A **1004**, subscribing client B **1006**, and subscribing client C **1008** may send the received event block object to event subscription device A **1024a**, event subscription device B **1024b**, and event subscription device C **1024c**, respectively.

ESPE **800** maintains the event block containership aspect of the received event blocks from when the event block is published into a source window and works its way through the directed graph defined by the one or more continuous queries **804** with the various event translations before being output to subscribers. Subscribers can correlate a group of subscribed events back to a group of published events by comparing the unique ID of the event block object that a publisher, such as publishing device **1022**, attached to the event block object with the event block ID received by the subscriber.

In an operation **916**, a determination is made concerning whether or not processing is stopped. If processing is not stopped, processing continues in operation **910** to continue receiving the one or more event streams containing event block objects from the, for example, one or more network devices. If processing is stopped, processing continues in an operation **918**. In operation **918**, the started projects are stopped. In operation **920**, the ESPE is shutdown.

As noted, in some embodiments, big data is processed for an analytics project after the data is received and stored. In other embodiments, distributed applications process continuously flowing data in real-time from distributed sources by applying queries to the data before distributing the data to geographically distributed recipients. As noted, an event stream processing engine (ESPE) may continuously apply the queries to the data as it is received and determines which entities receive the processed data. This allows for large amounts of data being received and/or collected in a variety of environments to be processed and distributed in real time. For example, as shown with respect to FIG. 2, data may be collected from network devices that may include devices within the internet of things, such as devices within a home automation network. However, such data may be collected from a variety of different resources in a variety of different

environments. In any such situation, embodiments of the present technology allow for real-time processing of such data.

Aspects of the current disclosure provide technical solutions to technical problems, such as computing problems that arise when an ESP device fails which results in a complete service interruption and potentially significant data loss. The data loss can be catastrophic when the streamed data is supporting mission critical operations such as those in support of an ongoing manufacturing or drilling operation. An embodiment of an ESP system achieves a rapid and seamless failover of ESPE running at the plurality of ESP devices without service interruption or data loss, thus significantly improving the reliability of an operational system that relies on the live or real-time processing of the data streams. The event publishing systems, the event subscribing systems, and each ESPE not executing at a failed ESP device are not aware of or affected by the failed ESP device. The ESP system may include thousands of event publishing systems and event subscribing systems. The ESP system keeps the failover logic and awareness within the boundaries of out-messaging network connector and out-messaging network device.

In one example embodiment, a system is provided to support a failover when event stream processing (ESP) event blocks. The system includes, but is not limited to, an out-messaging network device and a computing device. The computing device includes, but is not limited to, a processor and a computer-readable medium operably coupled to the processor. The processor is configured to execute an ESP engine (ESPE). The computer-readable medium has instructions stored thereon that, when executed by the processor, cause the computing device to support the failover. An event block object is received from the ESPE that includes a unique identifier. A first status of the computing device as active or standby is determined. When the first status is active, a second status of the computing device as newly active or not newly active is determined. Newly active is determined when the computing device is switched from a standby status to an active status. When the second status is newly active, a last published event block object identifier that uniquely identifies a last published event block object is determined. A next event block object is selected from a non-transitory computer-readable medium accessible by the computing device. The next event block object has an event block object identifier that is greater than the determined last published event block object identifier. The selected next event block object is published to an out-messaging network device. When the second status of the computing device is not newly active, the received event block object is published to the out-messaging network device. When the first status of the computing device is standby, the received event block object is stored in the non-transitory computer-readable medium.

FIG. 11 is a flow chart of an example of a process for generating and using a machine-learning model according to some aspects. Machine learning is a branch of artificial intelligence that relates to mathematical models that can learn from, categorize, and make predictions about data. Such mathematical models, which can be referred to as machine-learning models, can classify input data among two or more classes; cluster input data among two or more groups; predict a result based on input data; identify patterns or trends in input data; identify a distribution of input data in a space; or any combination of these. Examples of machine-learning models can include (i) neural networks; (ii) decision trees, such as classification trees and regression

trees; (iii) classifiers, such as Naïve bias classifiers, logistic regression classifiers, ridge regression classifiers, random forest classifiers, least absolute shrinkage and selector (LASSO) classifiers, and support vector machines; (iv) clusterers, such as k-means clusterers, mean-shift clusterers, and spectral clusterers; (v) factorizers, such as factorization machines, principal component analyzers and kernel principal component analyzers; and (vi) ensembles or other combinations of machine-learning models. In some examples, neural networks can include deep neural networks, feed-forward neural networks, recurrent neural networks, convolutional neural networks, radial basis function (RBF) neural networks, echo state neural networks, long short-term memory neural networks, bi-directional recurrent neural networks, gated neural networks, hierarchical recurrent neural networks, stochastic neural networks, modular neural networks, spiking neural networks, dynamic neural networks, cascading neural networks, neuro-fuzzy neural networks, or any combination of these.

Different machine-learning models may be used interchangeably to perform a task. Examples of tasks that can be performed at least partially using machine-learning models include various types of scoring; bioinformatics; cheminformatics; software engineering; fraud detection; customer segmentation; generating online recommendations; adaptive websites; determining customer lifetime value; search engines; placing advertisements in real time or near real time; classifying DNA sequences; affective computing; performing natural language processing and understanding; object recognition and computer vision; robotic locomotion; playing games; optimization and metaheuristics; detecting network intrusions; medical diagnosis and monitoring; or predicting when an asset, such as a machine, will need maintenance.

Any number and combination of tools can be used to create machine-learning models. Examples of tools for creating and managing machine-learning models can include SAS® Enterprise Miner, SAS® Rapid Predictive Modeler, and SAS® Model Manager, SAS Cloud Analytic Services (CAS)®, SAS Viya R of all which are by SAS Institute Inc. of Cary, North Carolina.

Machine-learning models can be constructed through an at least partially automated (e.g., with little or no human involvement) process called training. During training, input data can be iteratively supplied to a machine-learning model to enable the machine-learning model to identify patterns related to the input data or to identify relationships between the input data and output data. With training, the machine-learning model can be transformed from an untrained state to a trained state. Input data can be split into one or more training sets and one or more validation sets, and the training process may be repeated multiple times. The splitting may follow a k-fold cross-validation rule, a leave-one-out-rule, a leave-p-out rule, or a holdout rule. An overview of training and using a machine-learning model is described below with respect to the flow chart of FIG. 11.

In block 1102, training data is received. In some examples, the training data is received from a remote database or a local database, constructed from various subsets of data, or input by a user. The training data can be used in its raw form for training a machine-learning model or pre-processed into another form, which can then be used for training the machine-learning model. For example, the raw form of the training data can be smoothed, truncated, aggregated, clustered, or otherwise manipulated into another form, which can then be used for training the machine-learning model.

In block 1104, a machine-learning model is trained using the training data. The machine-learning model can be trained in a supervised, unsupervised, or semi-supervised manner. In supervised training, each input in the training data is correlated to a desired output. This desired output may be a scalar, a vector, or a different type of data structure such as text or an image. This may enable the machine-learning model to learn a mapping between the inputs and desired outputs. In unsupervised training, the training data includes inputs, but not desired outputs, so that the machine-learning model has to find structure in the inputs on its own. In semi-supervised training, only some of the inputs in the training data are correlated to desired outputs.

In block 1106, the machine-learning model is evaluated. For example, an evaluation dataset can be obtained, for example, via user input or from a database. The evaluation dataset can include inputs correlated to desired outputs. The inputs can be provided to the machine-learning model and the outputs from the machine-learning model can be compared to the desired outputs. If the outputs from the machine-learning model closely correspond with the desired outputs, the machine-learning model may have a high degree of accuracy. For example, if 90% or more of the outputs from the machine-learning model are the same as the desired outputs in the evaluation dataset, the machine-learning model may have a high degree of accuracy. Otherwise, the machine-learning model may have a low degree of accuracy. The 90% number is an example only. A realistic and desirable accuracy percentage is dependent on the problem and the data.

In some examples, if, at 1108, the machine-learning model has an inadequate degree of accuracy for a particular task, the process can return to block 1104, where the machine-learning model can be further trained using additional training data or otherwise modified to improve accuracy. However, if, at 1108, the machine-learning model has an adequate degree of accuracy for the particular task, the process can continue to block 1110.

In block 1110, new data is received. In some examples, the new data is received from a remote database or a local database, constructed from various subsets of data, or input by a user. The new data may be unknown to the machine-learning model. For example, the machine-learning model may not have previously processed or analyzed the new data.

In block 1112, the trained machine-learning model is used to analyze the new data and provide a result. For example, the new data can be provided as input to the trained machine-learning model. The trained machine-learning model can analyze the new data and provide a result that includes a classification of the new data into a particular class, a clustering of the new data into a particular group, a prediction based on the new data, or any combination of these.

In block 1114, the result is post-processed. For example, the result can be added to, multiplied with, or otherwise combined with other data as part of a job. As another example, the result can be transformed from a first format, such as a time series format, into another format, such as a count series format. Any number and combination of operations can be performed on the result during post-processing.

A more specific example of a machine-learning model is the neural network 1200 shown in FIG. 12. The neural network 1200 is represented as multiple layers of neurons 1208 that can exchange data between one another via connections 1255 that may be selectively instantiated thereamong. The layers include an input layer 1202 for receiving

input data provided at inputs 1222, one or more hidden layers 1204, and an output layer 1206 for providing a result at outputs 1277. The hidden layer(s) 1204 are referred to as hidden because they may not be directly observable or have their inputs or outputs directly accessible during the normal functioning of the neural network 1200. Although the neural network 1200 is shown as having a specific number of layers and neurons for exemplary purposes, the neural network 1200 can have any number and combination of layers, and each layer can have any number and combination of neurons.

The neurons 1208 and connections 1255 thereamong may have numeric weights, which can be tuned during training of the neural network 1200. For example, training data can be provided to at least the inputs 1222 to the input layer 1202 of the neural network 1200, and the neural network 1200 can use the training data to tune one or more numeric weights of the neural network 1200. In some examples, the neural network 1200 can be trained using backpropagation. Backpropagation can include determining a gradient of a particular numeric weight based on a difference between an actual output of the neural network 1200 at the outputs 1277 and a desired output of the neural network 1200. Based on the gradient, one or more numeric weights of the neural network 1200 can be updated to reduce the difference therebetween, thereby increasing the accuracy of the neural network 1200. This process can be repeated multiple times to train the neural network 1200. For example, this process can be repeated hundreds or thousands of times to train the neural network 1200.

In some examples, the neural network 1200 is a feed-forward neural network. In a feed-forward neural network, the connections 1255 are instantiated and/or weighted so that every neuron 1208 only propagates an output value to a subsequent layer of the neural network 1200. For example, data may only move one direction (forward) from one neuron 1208 to the next neuron 1208 in a feed-forward neural network. Such a “forward” direction may be defined as proceeding from the input layer 1202 through the one or more hidden layers 1204, and toward the output layer 1206.

In other examples, the neural network 1200 may be a recurrent neural network. A recurrent neural network can include one or more feedback loops among the connections 1255, thereby allowing data to propagate in both forward and backward through the neural network 1200. Such a “backward” direction may be defined as proceeding in the opposite direction of forward, such as from the output layer 1206 through the one or more hidden layers 1204, and toward the input layer 1202. This can allow for information to persist within the recurrent neural network. For example, a recurrent neural network can determine an output based at least partially on information that the recurrent neural network has seen before, giving the recurrent neural network the ability to use previous input to inform the output.

In some examples, the neural network 1200 operates by receiving a vector of numbers from one layer, transforming the vector of numbers into a new vector of numbers using a matrix of numeric weights, a nonlinearity, or both; and providing the new vector of numbers to a subsequent layer (“subsequent” in the sense of moving “forward”) of the neural network 1200. Each subsequent layer of the neural network 1200 can repeat this process until the neural network 1200 outputs a final result at the outputs 1277 of the output layer 1206. For example, the neural network 1200 can receive a vector of numbers at the inputs 1222 of the input layer 1202. The neural network 1200 can multiply the vector of numbers by a matrix of numeric weights to

determine a weighted vector. The matrix of numeric weights can be tuned during the training of the neural network 1200. The neural network 1200 can transform the weighted vector using a nonlinearity, such as a sigmoid tangent or the hyperbolic tangent. In some examples, the nonlinearity can include a rectified linear unit, which can be expressed using the equation $y=\max(x, 0)$ where y is the output and x is an input value from the weighted vector. The transformed output can be supplied to a subsequent layer (e.g., a hidden layer 1204) of the neural network 1200. The subsequent layer of the neural network 1200 can receive the transformed output, multiply the transformed output by a matrix of numeric weights and a nonlinearity, and provide the result to yet another layer of the neural network 1200 (e.g., another, subsequent, hidden layer 1204). This process continues until the neural network 1200 outputs a final result at the outputs 1277 of the output layer 1206.

As also depicted in FIG. 12, the neural network 1200 may be implemented either through the execution of the instructions of one or more routines 1244 by central processing units (CPUs), or through the use of one or more neuromorphic devices 1250 that incorporate a set of memristors (or other similar components) that each function to implement one of the neurons 1208 in hardware. Where multiple neuromorphic devices 1250 are used, they may be interconnected in a depth-wise manner to enable implementing neural networks with greater quantities of layers, and/or in a width-wise manner to enable implementing neural networks having greater quantities of neurons 1208 per layer.

The neuromorphic device 1250 may incorporate a storage interface 1299 by which neural network configuration data 1293 that is descriptive of various parameters and hyper parameters of the neural network 1200 may be stored and/or retrieved. More specifically, the neural network configuration data 1293 may include such parameters as weighting and/or biasing values derived through the training of the neural network 1200, as has been described. Alternatively, or additionally, the neural network configuration data 1293 may include such hyperparameters as the manner in which the neurons 1208 are to be interconnected (e.g., feed-forward or recurrent), the trigger function to be implemented within the neurons 1208, the quantity of layers and/or the overall quantity of the neurons 1208. The neural network configuration data 1293 may provide such information for more than one neuromorphic device 1250 where multiple ones have been interconnected to support larger neural networks.

Other examples of the present disclosure may include any number and combination of machine-learning models having any number and combination of characteristics. The machine-learning model(s) can be trained in a supervised, semi-supervised, or unsupervised manner, or any combination of these. The machine-learning model(s) can be implemented using a single computing device or multiple computing devices, such as the communications grid computing system 400 discussed above.

Implementing some examples of the present disclosure at least in part by using machine-learning models can reduce the total number of processing iterations, time, memory, electrical power, or any combination of these consumed by a computing device when analyzing data. For example, a neural network may more readily identify patterns in data than other approaches. This may enable the neural network to analyze the data using fewer processing cycles and less memory than other approaches, while obtaining a similar or greater level of accuracy.

Some machine-learning approaches may be more efficiently and speedily executed and processed with machine-learning specific processors (e.g., not a generic CPU). Such processors may also provide an energy savings when compared to generic CPUs. For example, some of these processors can include a graphical processing unit (GPU), an application-specific integrated circuit (ASIC), a field-programmable gate array (FPGA), an artificial intelligence (AI) accelerator, a neural computing core, a neural computing engine, a neural processing unit, a purpose-built chip architecture for deep learning, and/or some other machine-learning specific processor that implements a machine learning approach or one or more neural networks using semiconductor (e.g., silicon (Si), gallium arsenide (GaAs)) devices. These processors may also be employed in heterogeneous computing architectures with a number of and/or a variety of different types of cores, engines, nodes, and/or layers to achieve various energy efficiencies, processing speed improvements, data communication speed improvements, and/or data efficiency targets and improvements throughout various parts of the system when compared to a homogeneous computing architecture that employs CPUs for general purpose computing.

FIG. 13 illustrates various aspects of the use of containers 1336 as a mechanism to allocate processing, storage and/or other resources of a processing system 1300 to the performance of various analyses. More specifically, in a processing system 1300 that includes one or more node devices 1330 (e.g., the aforescribed grid system 400), the processing, storage and/or other resources of each node device 1330 may be allocated through the instantiation and/or maintenance of multiple containers 1336 within the node devices 1330 to support the performance(s) of one or more analyses. As each container 1336 is instantiated, predetermined amounts of processing, storage and/or other resources may be allocated thereto as part of creating an execution environment therein in which one or more executable routines 1334 may be executed to cause the performance of part or all of each analysis that is requested to be performed.

It may be that at least a subset of the containers 1336 are each allocated a similar combination and amounts of resources so that each is of a similar configuration with a similar range of capabilities, and therefore, are interchangeable. This may be done in embodiments in which it is desired to have at least such a subset of the containers 1336 already instantiated prior to the receipt of requests to perform analyses, and thus, prior to the specific resource requirements of each of those analyses being known.

Alternatively, or additionally, it may be that at least a subset of the containers 1336 are not instantiated until after the processing system 1300 receives requests to perform analyses where each request may include indications of the resources required for one of those analyses. Such information concerning resource requirements may then be used to guide the selection of resources and/or the amount of each resource allocated to each such container 1336. As a result, it may be that one or more of the containers 1336 are caused to have somewhat specialized configurations such that there may be differing types of containers to support the performance of different analyses and/or different portions of analyses.

It may be that the entirety of the logic of a requested analysis is implemented within a single executable routine 1334. In such embodiments, it may be that the entirety of that analysis is performed within a single container 1336 as that single executable routine 1334 is executed therein. However, it may be that such a single executable routine

1334, when executed, is at least intended to cause the instantiation of multiple instances of itself that are intended to be executed at least partially in parallel. This may result in the execution of multiple instances of such an executable routine 1334 within a single container 1336 and/or across multiple containers 1336.

Alternatively, or additionally, it may be that the logic of a requested analysis is implemented with multiple differing executable routines 1334. In such embodiments, it may be 10 that at least a subset of such differing executable routines 1334 are executed within a single container 1336. However, it may be that the execution of at least a subset of such differing executable routines 1334 is distributed across multiple containers 1336.

15 Where an executable routine 1334 of an analysis is under development, and/or is under scrutiny to confirm its functionality, it may be that the container 1336 within which that executable routine 1334 is to be executed is additionally configured assist in limiting and/or monitoring aspects of the 20 functionality of that executable routine 1334. More specifically, the execution environment provided by such a container 1336 may be configured to enforce limitations on accesses that are allowed to be made to memory and/or I/O addresses to control what storage locations and/or I/O devices may be accessible to that executable routine 1334. Such limitations may be derived based on comments within the programming code of the executable routine 1334 and/or other information that describes what functionality the 25 executable routine 1334 is expected to have, including what memory and/or I/O accesses are expected to be made when the executable routine 1334 is executed. Then, when the executable routine 1334 is executed within such a container 1336, the accesses that are attempted to be made by the executable routine 1334 may be monitored to identify any 30 behavior that deviates from what is expected.

Where the possibility exists that different executable routines 1334 may be written in different programming languages, it may be that different subsets of containers 1336 are configured to support different programming languages. 40 In such embodiments, it may be that each executable routine 1334 is analyzed to identify what programming language it is written in, and then what container 1336 is assigned to support the execution of that executable routine 1334 may be at least partially based on the identified programming 45 language. Where the possibility exists that a single requested analysis may be based on the execution of multiple executable routines 1334 that may each be written in a different programming language, it may be that at least a subset of the containers 1336 are configured to support the performance 50 of various data structure and/or data format conversion operations to enable a data object output by one executable routine 1334 written in one programming language to be accepted as an input to another executable routine 1334 written in another programming language.

55 As depicted, at least a subset of the containers 1336 may be instantiated within one or more VMs 1331 that may be instantiated within one or more node devices 1330. Thus, in some embodiments, it may be that the processing, storage and/or other resources of at least one node device 1330 may be partially allocated through the instantiation of one or more VMs 1331, and then in turn, may be further allocated within at least one VM 1331 through the instantiation of one or more containers 1336.

In some embodiments, it may be that such a nested allocation of resources may be carried out to affect an allocation of resources based on two differing criteria. By way of example, it may be that the instantiation of VMs

1331 is used to allocate the resources of a node device **1330** to multiple users or groups of users in accordance with any of a variety of service agreements by which amounts of processing, storage and/or other resources are paid for each such user or group of users. Then, within each VM **1331** or set of VMs **1331** that is allocated to a particular user or group of users, containers **1336** may be allocated to distribute the resources allocated to each VM **1331** among various analyses that are requested to be performed by that particular user or group of users.

As depicted, where the processing system **1300** includes more than one node device **1330**, the processing system **1300** may also include at least one control device **1350** within which one or more control routines **1354** may be executed to control various aspects of the use of the node device(s) **1330** to perform requested analyses. By way of example, it may be that at least one control routine **1354** implements logic to control the allocation of the processing, storage and/or other resources of each node device **1300** to each VM **1331** and/or container **1336** that is instantiated therein. Thus, it may be the control device(s) **1350** that effects a nested allocation of resources, such as the afore-described example allocation of resources based on two differing criteria.

As also depicted, the processing system **1300** may also include one or more distinct requesting devices **1370** from which requests to perform analyses may be received by the control device(s) **1350**. Thus, and by way of example, it may be that at least one control routine **1354** implements logic to monitor for the receipt of requests from authorized users and/or groups of users for various analyses to be performed using the processing, storage and/or other resources of the node device(s) **1330** of the processing system **1300**. The control device(s) **1350** may receive indications of the availability of resources, the status of the performances of analyses that are already underway, and/or still other status information from the node device(s) **1330** in response to polling, at a recurring interval of time, and/or in response to the occurrence of various preselected events. More specifically, the control device(s) **1350** may receive indications of status for each container **1336**, each VM **1331** and/or each node device **1330**. At least one control routine **1354** may implement logic that may use such information to select container(s) **1336**, VM(s) **1331** and/or node device(s) **1330** that are to be used in the execution of the executable routine(s) **1334** associated with each requested analysis.

As further depicted, in some embodiments, the one or more control routines **1354** may be executed within one or more containers **1356** and/or within one or more VMs **1351** that may be instantiated within the one or more control devices **1350**. It may be that multiple instances of one or more varieties of control routine **1354** may be executed within separate containers **1356**, within separate VMs **1351** and/or within separate control devices **1350** to better enable parallelized control over parallel performances of requested analyses, to provide improved redundancy against failures for such control functions, and/or to separate differing ones of the control routines **1354** that perform different functions. By way of example, it may be that multiple instances of a first variety of control routine **1354** that communicate with the requesting device(s) **1370** are executed in a first set of containers **1356** instantiated within a first VM **1351**, while multiple instances of a second variety of control routine **1354** that control the allocation of resources of the node device(s) **1330** are executed in a second set of containers **1356** instantiated within a second VM **1351**. It may be that the control of the allocation of resources for performing

requested analyses may include deriving an order of performance of portions of each requested analysis based on such factors as data dependencies thereamong, as well as allocating the use of containers **1336** in a manner that effectuates such a derived order of performance.

Where multiple instances of control routine **1354** are used to control the allocation of resources for performing requested analyses, such as the assignment of individual ones of the containers **1336** to be used in executing executable routines **1334** of each of multiple requested analyses, it may be that each requested analysis is assigned to be controlled by just one of the instances of control routine **1354**. This may be done as part of treating each requested analysis as one or more “ACID transactions” that each have the four properties of atomicity, consistency, isolation and durability such that a single instance of control routine **1354** is given full control over the entirety of each such transaction to better ensure that either all of each such transaction is either entirely performed or is entirely not performed. As will be familiar to those skilled in the art, allowing partial performances to occur may cause cache incoherencies and/or data corruption issues.

As additionally depicted, the control device(s) **1350** may communicate with the requesting device(s) **1370** and with the node device(s) **1330** through portions of a network **1399** extending thereamong. Again, such a network as the depicted network **1399** may be based on any of a variety of wired and/or wireless technologies and may employ any of a variety of protocols by which commands, status, data and/or still other varieties of information may be exchanged. It may be that one or more instances of a control routine **1354** cause the instantiation and maintenance of a web portal or other variety of portal that is based on any of a variety of communication protocols, etc. (e.g., a restful API). Through such a portal, requests for the performance of various analyses may be received from requesting device(s) **1370**, and/or the results of such requested analyses may be provided thereto. Alternatively, or additionally, it may be that one or more instances of a control routine **1354** cause the instantiation of and maintenance of a message passing interface and/or message queues. Through such an interface and/or queues, individual containers **1336** may each be assigned to execute at least one executable routine **1334** associated with a requested analysis to cause the performance of at least a portion of that analysis.

Although not specifically depicted, it may be that at least one control routine **1354** may include logic to implement a form of management of the containers **1336** based on the Kubernetes container management platform promulgated by Could Native Computing Foundation of San Francisco, CA, USA. In such embodiments, containers **1336** in which executable routines **1334** of requested analyses may be instantiated within “pods” (not specifically shown) in which other containers may also be instantiated for the execution of other supporting routines. Such supporting routines may cooperate with control routine(s) **1354** to implement a communications protocol with the control device(s) **1350** via the network **1399** (e.g., a message passing interface, one or more message queues, etc.). Alternatively, or additionally, such supporting routines may serve to provide access to one or more storage repositories (not specifically shown) in which at least data objects may be stored for use in performing the requested analyses.

Associated Processes

FIG. 14 illustrates one embodiment of method **1400** for securely executing analytical tasks using natural language. It shall be appreciated that other embodiments contemplated

within the scope of the present disclosure may involve more processes, fewer processes, different processes, or a different order of processes than illustrated in FIG. 14. It should be noted that a computer-program product may include a non-transitory machine-readable storage medium storing computer instructions that, when executed by one or more operations, may perform operations corresponding to the processes and sub-processes of method 1400. Additionally, or alternatively, a computer-implemented method may include operations corresponding to processes and sub-processes of 1400. Additionally, or alternatively, a computer-implemented system may include one or more processors, a memory, and a computer-readable medium operably coupled to the one or more processors, the computer-readable medium having computer-readable instructions stored thereon that, when executed by the one or more processors, cause a computing device to perform operations corresponding to the processes and sub-processes of method 1400.

One of ordinary skill in the art will appreciate that method 1400 may provide many technical advantages and practical applications over other techniques. For instance, other techniques may use flood prediction systems that are dependent on the presence of rivers or permanent bodies of water. However, such techniques may fail to account for regions that are prone to flash flooding or otherwise do not have rivers or permanent bodies of water. The techniques described herein may model flooding by simulating precipitation over a grid of cells within a region (e.g., rain-over-grid modeling), thus enabling these techniques to be used even in regions that lack rivers or other permanent bodies of water. Accordingly, the techniques described herein may enable a more reliable determination of sensor locations as compared to other flood prediction systems whose capabilities may be limited in at least some regions.

Additionally, method 1400 may utilize distributed parameters, accounting for unique characteristics of each cell of a grid of cells within a region (e.g., soil type, a percentage of imperviousness), rather than utilizing a single lumped parameter value for the entire region. Using distributed parameters may enable a flooding model to account for small-scale variations in water flow and accumulation within different areas, thus allowing for greater accuracy in determining sensor locations as compared to the lumped parameter approach.

Further, method 1400 may be specifically adapted to address dynamics of urban flooding that other flooding models may fail to address. For instance, urban environments may contain a high percentage of impervious surfaces which prevent water from infiltrating the ground, resulting in increased surface runoff. Other flooding models may fail to account for these impervious surfaces, whereas method 1400 may incorporate percentages and distributions associated with impermeable surfaces when determining locations for sensors. Accordingly, method 1400 may more accurately simulate flooding in regions with impervious surfaces as compared to these other techniques.

As shown in FIG. 14, process 1410 may create a geospatial dataset including features for a set of geographical cells within a target spatial area. A “geospatial dataset” may refer to a digital collection of data that defines location information (e.g., geographic information) for each geographical cell as well as corresponding features for the geographical cell. A “feature” may refer to a specific attribute or property of a geographical cell. A “geographical cell” may refer to a discrete, grid-like unit that represents a specific area of the Earth’s surface and a “set of geographical cells” may refer

to a collection of these discrete, grid-like units. A spatial area may refer to a region of the Earth’s surface and a target spatial area may be a spatial area defined by or including the area represented by the set of geographical cells.

In a non-limiting example, FIG. 15A may depict a geospatial dataset that includes features for a set of geographical cells within a target spatial area. For instance, FIG. 15A may depict geospatial dataset 1502, which may include N sets of features (e.g., features 1504A, features 1504B, features 1504C, and features 1504N). Each of the N sets of features may correspond to a respective geographical cell of the set of geographical cells, which may include at least geographical cells 1525A, 1525B, 1525C, 1525D, 1525E, 1525F, 1525G, 1525H, and 1525N. For instance, features 1504A may correspond to a first dataset entry 1505A for geographical cell 1525A, features 1504B may correspond to a second data entry 1505B for geographical cell 1525B, features 1504C may correspond to a third data entry 1505C for geographical cell 1525C, and features 1504N may correspond to an Nth data entry 1505N for geographical cell 1525N.

In some examples, creating the geospatial dataset may include obtaining a representation of the target spatial area and dividing the representation of the target spatial area into the set of geographical cells. The representation of the target spatial area may refer to geospatial data that defines bounds for the target spatial area. Each cell of the set of geographical cells may correspond to a unique (e.g., non-overlapping) area within the target spatial area and may have an equal size relative to each other geographical cell (e.g., the set of geographical cells may have a uniform size). Additionally, the set of geographic cells may be contiguous with each other. For instance, the set of geographic cells may be arranged in a grid-like formation. In a non-limiting example, as depicted in FIG. 15G-1, a representation 1554 of a target spatial area may be obtained and may be divided into a set of geographical cells. For instance, performing the dividing may generate grid 1556, where grid 1556 may be composed of or include geographical cells 1525. In some examples, the representation may include a raster, shapes files, or files in another geographic information system (GIS) file format.

Creating the geospatial dataset may further include process 1410 obtaining a set of feature maps associated with the target spatial area and extracting the features of the set of geographical cells from the set of feature maps. A “feature map” may refer to a collection of data that maps one or more points or sub-regions within a geospatial region to one or more features. Extracting the features of the set of geographical cells from the set of feature maps may include determining values for one or more features whose points or sub-regions map (e.g., align with, overlap) respective geographical cells of the set of geographical cells. In a non-limiting example, FIGS. 15G-2 and 15G-3 may depict various examples of feature maps (e.g., feature maps 1558, 1560, 1562, and 1564).

Additionally, creating the geospatial dataset may include process 1410 adding entries to the geospatial dataset that correspond to the set of geographical cells and that store the features extracted for the set of geographical cells. For instance, process 1410 may generate a data structure for each geographical cell of the set of geographical cells that represents the geographical cell and may populate the data structure with the extracted features relating to the geographical cell. Additionally, process 1410 may add each data structure as an entry to the geospatial dataset.

In some examples, the features for a respective geographical cell may include one or more identification features

associated with the respective geographical cell, where the one or more identification features may include an identification value associated with the geographical cell. An “identification feature” may refer to a unique attribute or property that may be used to differentiate one geographical cell from another geographical cell or groups of geographical cells from other groups of geographical cells. An “identification value” may refer to a value of an identification feature. The identification value may be generated by process 1410 when creating the geospatial dataset.

Additionally, or alternatively, the features for a respective geographical cell may include one or more topographical features associated with the respective geographical cell. A “topographical feature” may refer to an attribute or property that specifies spatial information about a geographical cell. For instance, the one or more topographical features may include an elevation of the respective geographical cell, a percentage of impervious surface within the respective geographical cell, a land cover classification associated with the respective geographical cell or a combination thereof. The elevation of a geographical cell may, in some examples, refer to an aggregate (e.g., average, median) height of the geographical cell relative to a fixed reference point (e.g., sea level). The percentage of impervious surface within a geographical cell may refer to a percentage of the area of the geographical cell that includes non-porous surfaces (e.g.,

or property that specifies information corresponding to soil composition, structure, or the movement or retention of water within a soil profile. For instance, the one or more soil and drainage features may include a soil type associated with the respective geographical cell, a type of storm drain present in the respective geographical cell, and a maximum capacity of the storm drain in the respective geographical cell. The soil type of a geographical cell may refer to a primary composition of soil within the geographical cell (e.g., a texture of the soil, a percentage of organic matter, a pH, a percentage of a particular element or chemical). The type of storm drain may refer to information corresponding to a location of a storm inlet (e.g., catch basin, storm drain inlet, curb inlet) as well as one or more stormwater infrastructure elements associated with the storm drain (e.g., inlets, outlets, pipes, culverts, stormwater channels). The maximum capacity of the storm drain may refer to the maximum duration and/or return period associated with a storm event that a storm drain is designed to handle.

In a non-limiting example, as depicted in FIG. 15G-2, feature map 1564 may be a feature map used to extract features related to a soil type of geographical cells. The various soil types associated with feature map 1564 may include those as defined by the Soil Survey Geographic Database (SSURGO). For instance, the various soil types may be depicted within the table below:

TABLE 1

SSURGO Soil Types	
Descriptor	Description
A	Deep, well drained sands or gravelly sands with high infiltration and low runoff rates
B	Deep well drained soils with a moderately fine to moderately coarse texture and a moderate rate of infiltration and runoff
C	Soils with a layer that impedes the downward movement of water or fine textured soils and a slow rate of infiltration
D	Soils with a very slow infiltration rate and high runoff potential. This group is composed of clays that have a high shrink-swell potential, soils with a high water table, soils that have a clay pan or clay layer at or near the surface, and soils that are shallow over nearly impervious material
A/D	Soils with a naturally very slow infiltration rate due to a high water table but will have high infiltration and low runoff rates if drained
B/D	Soils with a naturally very slow infiltration rate due to a high water table but will have a moderate rate of infiltration and runoff if drained
C/D	Soils with a naturally very slow infiltration rate due to a high water table but will have a slow rate of infiltration if drained

artificial surfaces, such as concrete and asphalt). The land cover classification within a geographical cell may refer to a land cover type associated with the geographical cell, where each land cover type may correspond to a different composition or arrangement of physical materials over the surface of the geographical cell (e.g., types of vegetation or ecosystems). In a non-limiting example, as depicted in FIGS. 15G-2 and 15-G-3, feature map 1558 may be a feature map used to extract features related to a percentage of impervious surfaces within geographical cells, feature map 1560 may be a feature map used to extract features related to an elevation within a geographical cell, and feature map 1562 may be a feature used to extract features related to a land cover classification of geographical cells.

Additionally, or alternatively, the features of the respective geographical cell may include one or more soil and drainage features associated with the respective geographical cell. A “soil or drainage feature” may refer to an attribute

As shown in FIG. 14, process 1420 may execute, via a flood simulation model, a computer simulation that simulates flooding within the target spatial area based on an input of the geospatial dataset and rainfall intensity data of one or more flooding events associated with the target spatial area. A “flood simulation model” may refer to a computational tool or module that predicts the effects of flooding. A “computer simulation” may refer to at least one instance of using the flood simulation to perform the prediction. “Rainfall intensity data” may refer to a collection of data that maps one or more points or sub-regions within a geospatial region to a quantity of rainfall amounts associated with the one or more points or sub-regions. The rainfall intensity data may be absolute (e.g., may include a total quantity of rainfall for each point or sub-region) or may be relative (e.g., may include a quantity or percentage of rainfall relative to a reference point or sub-region). “Flooding events” may be defined as a temporary overflow or inundation of water (e.g.,

due to rainfall, river overflow, storm surges, or snowmelt) to levels higher than a threshold amount typically associated with a geospatial area. The one or more flooding events (which may also be referred to as flood events) may be simulated flooding events (e.g., generated flooding events according to an algorithm) or may be historical flooding events (e.g., previously occurring flooding events whose data has been recorded). In some examples, the one or more flooding events may be of any duration and return period as obtained from National Oceanic and Atmospheric Administration (NOAA) Atlas 14 or may be based on projections, where the projections may be based on climate change.

It should be noted that, in some examples, flooding may be simulated based on storm surge data and/or sunny day flooding data. "Storm surge data" may refer to a collection of data that maps one or more points or sub-regions within a geospatial region to a respective water level rise caused by an associated storm in examples in which the geospatial region is a coastal region. "Sunny day flooding data" may refer to a collection of data that maps one or more points or sub-regions within a geospatial region to a respective water level rise caused by high tide flooding (e.g., via water coming inland from the coast through drainpipes) or by backflow from another body of water (e.g., backflow from rivers, lakes, creeks, or other bodies of water). Flooding events that have corresponding rainfall intensity data and one or more of storm surge data or sunny day flooding data may be referred to as a compound flooding event. It should be noted that, in some examples, there may be flooding events that only have storm surge data and/or only have sunny day flooding data.

In a non-limiting example, FIG. 15A may depict executing, via a flood simulation model, a computer simulation that simulates flooding within the target spatial area based on an input of the geospatial dataset and rainfall intensity data of one or more flooding events associated with the target spatial area. For instance, FIG. 15A may depict rainfall intensity data 1506, which may include rainfall data sets for each of N flood events (e.g., dataset 1507A for a first flood event 1508A, dataset 1507B for a second flood event 1508B, dataset 1507C for a third flood event 1508C, dataset 1507N for an Nth flood event 1508N). The rainfall intensity data 1506 and geospatial dataset 1502 may be provided as an input 1509 to flood simulation model 1510. A computer simulation 1511 may be executed via flood simulation model 1510. It should be noted that the number of datasets and/or the number of flood events associated with rainfall intensity data 1506 may, in some examples, be the same as or differ from the number of features associated with geospatial dataset 1502.

As shown in FIG. 14, sub-process 1420A of process 1420 may simulate inundation levels of the set of geographical cells during the one or more flooding events. An "inundation level" may refer to a depth or height of water covering a region. An inundation level may be determined for each of the set of geographical cells and for each of the one or more flooding events, where the inundation level for a particular geographical cell and a particular flooding event may refer to a depth or height of water covering an area associated with the particular geographical cell for the particular flooding event.

In a non-limiting example, FIG. 15A may depict simulating inundation levels of the set of geographical cells during the one or more flooding events. For instance, executing computer simulation 1511 may simulate inundation levels for each geographical cell for each flooding event. For instance, executing computer simulation 1511

may simulate inundation levels 1512A for a first flood event (e.g., flood event 1508A) for a first geographical cell (e.g., geographical cell 1525A), inundation levels 1512B for a second flood event (e.g., flood event 1508B) for the first geographical cell, inundation levels 1512C for a third flood event (e.g., flood event 1508C) for the first geographical cell, and inundation levels 1512N for an Nth flood event (e.g., flood event 1508N) for the first geographical cell. Additionally, executing computer simulation 1511 may 10 simulate inundation levels 1514A, 1514B, 1514C, and 1514N for the first, second, third, and Nth flood events, respectively, for a second geographical cell (e.g., geographical cell 1525B); inundation levels 1516A, 1516B, 1516C, and 1516N for the first, second, third, and Nth flood events, respectively, for a third geographical cell (e.g., geographical cell 1525C); and inundation levels 1518A, 1518B, 1518C, and 1518N for the first, second, third, and Nth flood events of an Nth geographical cell (e.g., geographical cell 1525N). It should be noted that there may be examples in which the 20 number of flood events and number of geographical cells may differ (e.g., N geographical cells and M flood events) without deviating from the scope of the present disclosure.

In some examples, simulating the inundation levels of the 25 set of geographical cells during a respective flooding event may include sub-process 1420A identifying, from the rainfall intensity data, precipitation amounts observed during a set of timesteps of the respective flooding event and simulating, via the simulation model, the precipitations amounts observed during the set of timesteps on the set of geographical cells. For instance, in a non-limiting example, FIG. 15H 30 may represent rainfall intensity data within a dataset 1507 for a particular flooding event. The rainfall intensity data may include a set of timesteps (e.g., 9:21:00 AM, 9:36:00 AM) and precipitation quantities associated with each timestep. The precipitation quantities may include a total amount of precipitation that occurs between the timestep and a subsequent timestep (e.g., between 9:21:00 AM and 9:36:00 AM) or between the timestep and a previous timestep (e.g., between 9:07:00 AM and 9:21:00 AM). Additionally, 35 the precipitation quantities may include a cumulative precipitation tracking a total amount of precipitation that has fallen since an initial timestep. For each timestep, sub-process 1420A may use the total amount of precipitation for the timestep on each geographical cell of the set of geographical cells as the simulated precipitation amount for the timestep (e.g., each geographical cell may have a same precipitation for a given timestep). It should be noted that the timesteps associated with the rainfall intensity data (e.g., timesteps of 14 minutes) may differ in granularity, in some 40 examples, from the timesteps at which an inundation level is computed (e.g., timesteps of 30 seconds). In some such examples, multiple timesteps at which the inundation level is computed may map to a single timestep associated with the rainfall intensity data or vice-versa. It should be noted that 45 there may be examples in which the inundation levels are simulated using runoff and/or infiltration data along with the rainfall intensity data. Additionally, or alternatively, without deviating from the scope of the disclosure, it should be noted that inundation levels may be further obtained by observing high-water marks, analyzing 311 calls, and/or processing photographs or videos (e.g., photographs or videos associated with a flood event).

Using the simulated precipitation amounts, sub-process 1420A 50 may compute an inundation level for each timestep of a flood event for a geographical cell. In a non-limiting example, FIG. 15D-1 may depict computed inundation levels for each timestep of a flood event for a set of

geographical cells. For instance, the inundation levels **1512A** for a first flood event (e.g., flood event **1508A**) of a first geographical cell (e.g., geographical cell **1525A**) may include inundation level **1512AA** for a first timestep, inundation level **1512BB** for a second timestep, inundation level **1512CC** for a third timestep, and inundation level **1512NN** for an Nth timestep. Additionally, for the first flood event, inundation levels **1514A** of a second geographical cell (e.g., geographical cell **1525B**) may include inundation levels **1514AA**, **1514BB**, **1514CC**, and **1514NN** for the first, second, third, and Nth timesteps, respectively; inundation levels **1516A** of a third geographical cell (e.g., geographical cell **1525C**) may include inundation levels **1516AA**, **1516BB**, **1516CC**, and **1516NN** for the first, second, third, and Nth timesteps, respectively; and inundation levels **1518A** of an Nth geographical cell (e.g., geographical cell **1525N**) may include inundation levels **1518AA**, **1518BB**, **1518CC**, and **1518NN** for the first, second, third, and Nth timesteps, respectively. It should be noted that the number of timesteps and the number of geographical cells may differ (e.g., N geographical cells and P timesteps, where both N and P are integers) without deviating from the scope of the present disclosure.

Computing the inundation level for a geographical cell may include identifying and/or accounting for one or more of a precipitation amount during a respective timestep (e.g., a total amount of precipitation that occurs between a timestep and a subsequent or previous timestep), an amount of water flowing from other geographical cells into the respective geographical cell during the respective timestep, an amount of water flowing from the respective geographical cell to the other geographical cells during the respective timestep, an amount of water lost due to infiltration during the respective timestep, and an amount of water removed via storm drain outflow during the respective timestep. Additionally, or alternatively, computing the inundation level may include identifying and/or accounting for water flow into and out of a geographical cell due to a storm surge, waves overtopping barriers, and/or water moving inland (e.g., from the ocean). Using some or each of these quantities, sub-process **1420A** may compute the inundation level of the respective geographical cell. For instance, sub-process **1420A** may sum the precipitation amount observed during the respective timestep with the amount of water flowing from other geographical cells into the respective geographical cell. Additionally, sub-process **1420A** may subtract the amount of water flowing from the respective geographical cell to the other geographical cells, the amount of water lost due to infiltration, and the amount of water removed via storm drain outflow from a result of the summing. It should be noted that there may be examples where inundation levels are determined based on corresponding storm surge data and/or sunny day flooding data.

In some examples, sub-process **1420A** may determine a maximum inundation level per geographical cell over the set of timesteps for a given flood event (e.g., a per-cell maximum inundation level). For instance, in a non-limiting example as depicted in FIG. 15D-1, per-cell maximum inundation level detection process **1526** may detect that inundation level **1512NN** associated with an Nth timestep is a maximum inundation level relative to each other inundation level of the inundation levels **1512A** (e.g., inundation levels **1512AA**, **1512BB**, and **1512CC**). Similarly, per-cell maximum inundation level detection process **1526** may detect that inundation level **1514BB** associated with a second timestep is a maximum inundation level relative to each other inundation level of the inundation levels **1514A** (e.g.,

inundation levels **1514AA**, **1514CC**, and **1514NN**); that inundation level **1516CC** associated with a third timestep is a maximum inundation level relative to each other inundation level of the inundation levels **1516A** (inundation levels **1516AA**, **1516BB**, and **1516NN**); and that inundation level **1518AA** associated with a first timestep is a maximum inundation level relative to each other inundation level of the inundation levels **1518A** (e.g., inundation levels **1518BB**, **1518CC**, and **1518NN**).

10 Additionally, sub-process **1420A** may determine an overall maximum inundation level (e.g., to be used as a normalizing divisor in later steps). For instance, in a non-limiting example as depicted in FIG. 15D-1, overall maximum inundation level detection process **1528** may determine that inundation level **1518AA** of inundation levels **1518A** may have a maximum inundation level relative to each other inundation level for the set of geographical cells for a particular flood event (e.g., a higher value than inundation level **1512NN** of inundation levels **1512A**, inundation level **1514BB** of inundation levels **1514A**, and inundation level **1516CC** of inundation levels **1516A**).

As shown in FIG. 14, sub-process **1420B** of process **1420** may simulate a flow of water across the set of geographical cells during the one or more flooding events. A flow of water as described herein may refer to a metric that specifies a direction of water movement between a geographical cell and another location (e.g., between one geographical cell and another geographical cell or between a geographical cell and a region outside the target spatial area). In some examples, the metric may further describe a magnitude associated with the movement of water between the geographical cell and the other location.

25 In a non-limiting example, FIG. 15A may depict simulating a flow of water across the set of geographical cells during the one or more flooding events (e.g., via computer simulation **1511**). For instance, the set of geographical cells may include geographical cells **1525A** through **1525N**. Executing computer simulation **1511** may result in determining a flow is present from geographical cell **1525E** to geographical cell **1525D**; from geographical cell **1525D** to **1525N**; from geographical cell **1525F** to geographical cell **1525A**; from geographical cell **1525G** to **1525A**; from geographical cell **1525A** to geographical cell **1525N**; from geographical cell **1525H** to geographical cell **1525B**; and from geographical cell **1525B** to geographical cell **1525C**.

The flow of water across the set of geographical cells may be represented using a data structure, such as a graph data structure, which may be referred to as a water flow network dataset or a from-to dataset. The graph data structure may include a set of nodes corresponding to the set of geographical cells and a set of edges specifying water flow trajectories among the set of nodes. For instance, in a non-limiting example, geographical cells **1525E** and **1525D** may each map to a respective node of the set of nodes and the water flow between them (e.g., from geographical cell **1525E** and **1525D**) may map to a respective edge of the set of edges (e.g., a directional edge). In some examples, a respective edge extending from a first respective node to a second respective node may specify a water flow trajectory (e.g., a water flow direction) that indicates water moves from a geographical cell corresponding to the first respective node to a geographical cell corresponding to the second respective node during flooding (e.g., from geographical cell **1525E** to geographical cell **1525D**).

50 As shown in FIG. 14, process **1430** may convert the inundation levels of the set of geographical cells to normalized inundation scores for the set of geographical cells.

Converting the inundation levels to a normalized inundation scores may refer to converting computed inundation levels, whose lower or upper bounds may vary, to a common scale with fixed lower or upper bounds (e.g., with bounds such that a highest normalized inundation score is 1). For instance, an overall maximum inundation level among each inundation level associated with a flood event for the set of geographical cells may be determined. Converting the inundation levels to normalized inundation scores in such examples may include dividing inundation levels for each geographical cell (e.g., a maximum inundation level for each geographical cell) by the determined overall maximum inundation level.

In a non-limiting example, FIGS. 15D-1 and 15D-2 may depict converting inundation levels to normalized inundation scores. For instance, as described herein, per-cell maximum inundation level detection process 1526 may detect a maximum inundation level per geographical cell for a flood event (e.g., inundation levels 1512NN, 1514BB, 1516CC, and 1518AA for flood event 1508A) and overall maximum inundation level detection process may detect an overall maximum inundation level for the set of geographical cells for the flood event (e.g., inundation level 1518AA for flood event 1508A). Inundation level normalization process 1530 may determine normalized inundation scores corresponding to the flood event for each of the set of geographical cells. For instance, inundation level normalization process 1530 may determine a first normalized inundation score 1532A for a first geographical cell (e.g., geographical cell 1525A), a second normalized inundation score 1534A for a second geographical cell (e.g., geographical cell 1525B), a third normalized inundation score 1536A for a third geographical cell (e.g., geographical cell 1525C), and an Nth normalized inundation 1538A score for an Nth geographical cell (e.g., geographical cell 1525N). To perform normalization, inundation level normalization process 1530 may divide the maximum per cell inundation value by the overall maximum inundation level. For instance, normalized inundation score 1532A may be calculated by dividing inundation level 1512NN by inundation level 1518AA, normalized inundation score 1534A may be calculated by dividing inundation level 1514BB by inundation level 1518AA, normalized inundation score 1536A may be calculated by dividing inundation level 1516CC by inundation level 1518AA, and normalized inundation score 1538A may be calculated by dividing inundation level 1518AA by itself (e.g., and may thus have a value of 1).

Inundation score normalization may be performed for each flood event to generate sets of normalized inundation scores for each flood event for each geographical cell. For instance, as depicted in FIG. 15E, normalized inundation scores 1532A, 1534A, 1536A, and 1538A may be computed for first, second, third, and Nth geographical cells (e.g., geographical cells 1525A, 1525B, 1525C, and 1525N, respectively) for a first flood event (e.g., flood event 1508A) using the techniques described herein (e.g., as depicted with reference to FIGS. 15D-1 and 15D-2). Similar techniques may be performed to obtain normalized inundation scores 1532B, 1532C, and 1532N for the first geographical cell for second, third, and Nth flood events; normalized inundation scores 1534B, 1534C, and 1534N for the second geographical cell for the second, third, and Nth flood events; normalized inundation scores 1536B, 1536C, and 1536N for the third geographical cell for the second, third and Nth flood events; and normalized inundation scores 1538B, 1538C, and 1538N for the fourth geographical cell for the second, third, and Nth flood events. It should be noted that the

number of flood events (and thus the number of normalized inundation scores per cell) may be different than the number of geographical cells (e.g., M flood events) without deviating from the scope of the present disclosure.

The normalized inundation scores for each geographical cell may be aggregated into an overall normalized inundation score for the flood events. For instance, as depicted in FIG. 15E, normalized inundation score aggregation process 1540 may aggregate (e.g., calculate the mean, median, or maximum) of normalized inundation scores 1532A, 1532B, 1532C, and 1532N into overall normalized inundation score 1542A for the first geographical cell (e.g., geographical cell 1525A); normalized inundation scores 1534A, 1534B, 1534C, and 1534N into overall normalized inundation score 1542B for the second geographical cell (e.g., geographical cell 1525B), normalized inundation scores 1536A, 1536B, 1536C, and 1536N into overall normalized inundation score 1542C for the third geographical cell (e.g., geographical cell 1525C), and normalized inundation scores 1538A, 1538B, 1538C, and 1538N into overall normalized inundation score 1542N for the Nth geographical cell (e.g., geographical cell 1525N).

As shown in FIG. 14, process 1440 may detect one or more clusters of interconnected geographical cells based on the simulating of the flow of water. The geographical cells being interconnected may refer to each geographical cell within a particular cluster having a water flow path (e.g., as determined by sub-process 1420A) that connects it to another geographical cell of the cluster. For instance, in a non-limiting example, as depicted in FIG. 15A, a flow of water graph 1520 may be provided via computer simulation 1511. In such examples, geographical cells 1525A, 1525D, 1525E, 1525F, 1525G, and 1525N may be included within a first cluster and geographical cells 1525B, 1525C, and 1525H may be included within a second cluster. Each geographical cell within the first cluster may be isolated from the second cluster. For instance, geographical cells 1525B, 1525C, and 1525H may not have any water flow to or from geographical cells of the first cluster and geographical cells 1525A, 1525D, 1525E, 1525F, 1525G, and 1525N may not have any water flow to or from geographical cells of the second cluster.

Each cluster, in some examples, may be referred to as a connected component, where a connected component may be a graph of a set of nodes (e.g., geographical cells) that are reachable from each other (e.g., a set of vertices in a graph that are linked to each other by paths). If all nodes are reachable while accounting for direction from at least one node, the connected component may be referred to as a strongly connected component (e.g., all geographical cells of the second cluster may be accessible from geographical cell 1525H). If some nodes are only reachable when ignoring direction, then the connected component may be referred to as a weakly connected component (e.g., graphical cell 1525E only has a path to geographical cell 1525G of the first cluster when ignoring water flow direction).

In some examples, a respective cluster of geographical cells may correspond to a respective sub-catchment (e.g., a respective independent sub-area) within the target spatial area, where a sub-catchment may refer to a region where surface runoff within the region is directed to a single discharge point (e.g., a single geographical cell) and/or to a region in which water circulates. For instance, from any geographical cell of the first cluster of FIG. 15A (e.g., 1525A, 1525D, 1525E, 1525F, 1525G, and 1525N), water may flow to geographical cell 1525N, which may represent or may include a discharge point. Similarly, from any

geographical cell of the second cluster of FIG. 15A (e.g., **1525B**, **1525C**, and **1525H**), water may flow to geographical cell **1525C**.

Each cluster of interconnected cells may indicate respective flow characteristics and/or a respective retention size for the sub-catchment corresponding to the cluster. A “flow characteristic” may refer to a specific property of water movement within the sub-catchment (e.g., flow rate, direction, timing, volume). A “retention size” may refer to a capacity of the sub-catchment to hold or store water (e.g., for a temporary time), which may reduce runoff.

To construct a cluster, process **1440** may locate a data structure that includes representations of the geographical cells and the water flow between them (e.g., a water flow direction graph as generated by sub-process **1420B**). For instance, if the data structure is a graph, the graph data structure, the graph data structure may include objects representing nodes (e.g., geographical cells) and objects representing edges connecting nodes (e.g., waterflow between nodes). Upon locating the data structure, process **1440** may identify that each of a first set of nodes of the graph data structure are connected via a first set of edges and may construct a cluster corresponding to the first set of nodes. Likewise, process **1440** may identify that each of a second set of nodes are connected via a second set of edges. If at least one edge of the second set of nodes connects to a node from the first set of nodes (e.g., if the first set of nodes and second set of nodes are not mutually exclusive), process **1440** may add the second set of nodes to the cluster. However, if there are not any edges of the second set of edges connecting nodes from the first set of nodes to nodes of the second set of nodes (e.g., if the first set of nodes and second set of nodes are disjoint), process **1440** may construct another cluster for the second set of nodes.

In a non-limiting example, FIG. 15B may depict an example of performing clustering. For instance, a flow of water graph **1520** (e.g., a water flow direction graph) may be provided as an input to an interconnected cells detector **1522**, which may identify, using the techniques as described herein, sets of geographical cell clusters. For instance, interconnected cells detector **1522** may identify geographical cell clusters **1524A**, **1524B**, **1524C**, **1524D**, **1524E**, **1524F**, and **1524N**, each of which may correspond to a unique subset of the set of geographical cells.

FIG. 15C may provide a visual depiction of the detected clusters as described with reference to FIG. 15B. For instance, FIG. 15C may depict a grid of geographical cells **1525**, where the grid of geographical cells **1525** may be divided into clusters **1524A**, **1524B**, **1524C**, **1524D**, **1524E**, **1524F**, and **1524N**. In some examples, only geographical cells **1525** located at least partially over land may be included in the clustering. Additionally, or alternatively, geographical cells **1525** located completely within water (e.g., the ocean) may not be included in clustering.

As shown in FIG. 14, process **1450** may determine, via an optimization algorithm, a set of optimal sensor locations in the target spatial area based on the normalized inundation scores and the one or more clusters of interconnected geographical cells. An “optimization algorithm” may refer to a computation method used to locate a solution that maximizes or minimizes a specific objective function under given constraints. A “sensor”, which may also be called a flood sensor or a water level sensor, may be a device configured to detect the presence, depth, and/or flow of water in a particular area to monitor and alert for potential flooding conditions. The placement of sensors may be such

that a direct or indirect estimation of water level in the target spatial area may be determined via readings from the sensors.

Determining a set of optimal sensor locations via an optimization algorithm may refer to using an optimization algorithm that identifies sensor locations such that they maximize a sum of cell inundation scores for geographical cells with sensors and/or maximize a number of connected components with at least one sensor location. Additionally, or alternatively, determining a set of optimal sensor locations via an optimization algorithm may refer to using an optimization algorithm that identifies sensor locations such that the sensor locations correspond to locations that have a greatest inundation score (e.g., flood most frequently or are hotspots), locations which are highly connected (e.g., within clusters with a high number of nodes relative to other clusters), or locations which are identified as having a critical status. In some examples, the optimization algorithm may include a constraint that constrains the set of optimal sensor locations to a pre-defined number of optimal sensor locations. For instance, the optimization algorithm may use a fixed number of sensors according to the constraint.

As a non-limiting example, FIG. 15F may depict overall normalized inundation scores **1538A** through **1538N** and geographical cell clusters **1524A** through **1524N** being provided to optimization algorithm **1548**. Optimization algorithm **1548** may be used to determine a set of geographical cells within which a sensor should be located (e.g., each geospatial marker **1550A** through **1550RR** may indicate that an overlapping geographical cell **1525** includes a sensor). In some examples, optimization algorithm **1548** may have sensor quantity constraint **1553**, which may specify a fixed number of sensors and/or a fixed number of sensor points or sensor locations.

In some examples, process **1450** may further determine, via the optimization algorithm, the set of optimal sensor locations in the target spatial area based on a criticality status (which may also be referred to as a critical status) of each geographical cell of the set of geographical cells. The criticality status may indicate whether a given geographical cell is associated with a given level of functionality that may be disrupted if a flooding event were to occur. In a non-limiting example, as depicted in FIG. 15F, IDs of critical geographical cells **1544** (e.g., cells with a positive critical status) may be provided to optimization algorithm **1548**. Additionally, a flood amplification factor **1545** for critical geographical cells **1544** may be provided to optimization algorithm **1548**, where the flood amplification factor may provide a weight to the critical status of a geographical cell in the optimization algorithm (e.g., a higher flood amplification factor may provide a greater weight). The flood amplification factor **1545** may be configured via input **1546** (e.g., user input to a user interface) or may be pre-configured. In some examples, optimization algorithm may include a constraint (e.g., critical location inclusion constraint **1551**) that constrains the set of optimal sensor locations to include one or more pre-determined critical locations (e.g., the critical geographical cells **1544** whose IDs are provided to optimization algorithm **1548**). In some examples, geographical cells with a positive critical status may include those including flood control and response elements (e.g., pumping stations, floodgates, fire stations, government buildings, ramps to evacuation roads, or major highways), high population density areas which may need early evacuation (e.g., office buildings, malls, schools), or locations that meet additional criteria provided by a governmental entity.

In some examples, the optimization function may maximize a multi-criterion objective function. A non-limiting example of the multi-criterion objective function for the optimization function may be given as $W_{in} * E_{i \text{ in } c} w_i * In_i * x_i + W_{cc} * \sum_j y_j$. W_{in} may refer to a weight of an inundation coverage objective. C may refer to a set of geographical cells (e.g., geographical cells 1525) and i may refer to an ith geographical cell within C. w_i may refer to a variable that has a value m (e.g., amplification factor 1545) if the ith geographical cell has a critical status (e.g., if the geographical cell is one of the critical geographical cells 1544). In_i may refer to a normalized inundation score (e.g., one of the overall normalized scores 1538A through 1538N) for the ith geographical cell. x_i may be a decision variable that may have a value of 1 if a sensor is located in the ith geographical cell and may have a 0 otherwise. W_{cc} may be a weight of a connected component coverage objective. CC may refer to a set of connected components (e.g., geographical cell clusters 1524A through 1524N) and j may refer to a jth connected component within CC. y_j may be a decision variable that may have a value of 1 if the connected component is covered for at least one sensor (e.g., at least one geographical cell within the connected component has a sensor) and may be 0 otherwise. In some examples, the multi-criterion objective function may have the constraints that $\sum_i x_i = N$, where N represents a total number of sensors, and $\sum_i x_i \geq y_j$ for each j in CC, where $x_i \in \{0,1\}$ for each i in C and $y_j \in \{0,1\}$ for each j in CC. The optimization function may attempt to maximize the multi-criterion objective function. The sum of W_{cc} and W_{in} may add up to 1.

In some examples, a multi-criterion objective function may be a weighted sum of an inundation coverage objective function and a cluster coverage objective function. For instance, $W_{in} * \sum_i w_i * In_i * x_i$ may be an example of an inundation coverage objective function, $W_{cc} * \sum_j y_j$ may be an example of a cluster coverage objective function, and $W_{in} * \sum_i w_i * In_i * x_i + W_{cc} * \sum_j y_j$ may represent a weighted sum of these functions. The inundation coverage objective function may prioritize sensor locations in geographical cells that yield a highest aggregated normalized inundation score (e.g., associated with a highest In_i) across different combinations of the set of graphical cells and the cluster coverage objective function may prioritize sensor locations in geographical cells that maximize coverage across the one or more clusters of interconnected geographical cells (e.g., arrangements resulting in the highest number of clusters with at least one sensor). In some examples, the inundation coverage objective function may prioritize the sensor locations in the geographical cells that are assigned to a criticality multiplier value (e.g., w_i) due to a presence of a pre-defined type of asset (e.g., a critical status) and that yield the highest aggregated normalized inundation score across different combinations of the set of geographical cells.

It should be noted that there may be examples (e.g., examples in which flooding events include sunny day flooding data) in which the set of optimal sensor locations may be further based on a location of one or more inlets (e.g., one or more inlets that may flood due to sunny day flooding). For instance, the optimization algorithm may account for the location of one or more storm inlets (e.g., may be constrained to include a number of one or more storm inlets) when being used to determine the set of optimal sensor locations. Additionally, or alternatively, the optimization algorithm may give higher priority to geographical cells within a predefined threshold distance from a coast (e.g., 100 feet, 500 feet, 1000 feet). Sensors that are placed within

storm inlets may be utilized for determining a velocity of water for geographical cells within a certain distance (e.g., to determine a relationship between the distance of an inlet to the ocean and a corresponding velocity of water).

In some examples, the optimization algorithm may identify sets of geographical cells that may have at most one sensor and may accordingly ensure that these geographical cells have no more than one sensor placed there. Additionally, the optimization algorithm may identify prohibited geographical cells for sensors and may refrain from placing any sensors at the prohibited geographical cells. Additionally, the optimization algorithm may identify geographical cells to be automatically selected (e.g., according to a constraint indicating that these geographical cells are required to have a sensor) and may automatically include these geographical cells in those that include a sensor.

As shown in FIG. 14, process 1460 may generate a sensor location map that includes the set of geographical cells and a set of geospatial markers identifying the set of optimal sensor locations. A "sensor location map" may refer to a digital representation of an arrangement of the set of optimal sensor locations relative to the set of geographical cells. A "geospatial marker" may refer to a digital indicator that marks a respective optimal sensor location relative to a respective geospatial cell. In some examples, generating the sensor location map may include overlaying the set of geospatial markers on the set of geographical cells. For instance, each geospatial marker of the set of geospatial markers may appear to overlap a corresponding geographical cell of the set of geographical cells.

In a non-limiting example, as depicted in FIG. 15F, a sensor location map 1552 may include a set of geographical cells 1525 and a set of geospatial markers (e.g., geospatial markers 1550A through 1550RR). At least some of the set of geographical cells 1525 may belong to a respective cluster of a set of clusters, where some or each of these clusters may have at least one sensor. For instance, each of geospatial markers, 1550A, 1550B, 1550C, 1550D, 1550E, 1550F, 1550G, 1550H, 1550I, 1550J, 1550K, 1550L, 1550M, 1550N, 1550O, 1550P, 1550Q, 1550R, 1550S, 1550T, 1550U, 1550V, 1550W, 1550X, 1550Y, 1550Z, 1550AA, 1550BB, 1550CC, 1550DD, 1550EE, 1550FF, 1550GG, 1550HH, 1550II, 1550JJ, 1550KK, 1550LL, 1550MM, 1550NN, 1550OO, 1550PP, 1550QQ, and 1550RR may belong to one of geographical cell clusters 1524A through 1524N (e.g., as depicted with reference to FIG. 15B).

As shown in FIG. 14, process 1470 may generate a visual output that displays the sensor location map after generating the sensor location map. A "visual output that displays the sensor location map" may refer to a graphical representation of the information associated with the sensor location map (e.g., the sensor locations and their positions relative to the set of geographical cells).

In some examples, generating the sensor location map may include overlaying the set of geospatial markers on the set of geographical cells and generating the visual output may include displaying the sensor location map in a geographical user interface after generating the sensor location map. For instance, in a non-limiting example, sensor map 1552 of FIG. 14F may be provided to a graphical user interface and may be displayed as visually depicted in FIG. 14F. Alternatively, the sensor map may have a different appearance (e.g., different widths or heights of geographical cells 1525 or different shapes or colors for the geographical markers). In some examples, displaying the sensor location map may include transmitting one or more data packets that

51

include the sensor location map from a system that determines the optimal sensor locations to a device that displays the graphical user interface.

Additionally, or alternatively, generating the visual output may include generating a report that includes a visual depiction of the sensor location map after generating the sensor location map. A “report” may refer to a document (e.g., a digital artifact or a printed document) that is generated based on generating the sensor location map. A “visual depiction of the sensor location map” may refer to a graphical representation of the information associated with the sensor location map (e.g., the sensor locations and their positions relative to the set of geographical cells).

In some examples, one or more sensors may be positioned at the optimal sensor locations based on the visual output that displays the sensor location map. In some examples, positioning the one or more sensors may include a system (e.g., the system that generated the visual output) providing an indication (e.g., to a user interface, to a database, to another system, to a digital artifact stored by the system) to store or set position information (e.g., geographical information) for the one or more sensors. For instance, after the system generates a visual output, an input to an application (e.g., a geographic information system (GIS)) that sets the positions of the one or more sensors may be provided (e.g., via a user or via an automated process).

It shall also be noted that the system and methods of the embodiments and variations described herein can be embodied and/or implemented at least in part as a machine comprising a computer-readable medium storing computer-readable instructions. The instructions may be executed by computer-executable components integrated with the system and one or more portions of the processors and/or the controllers. The computer-readable medium can be stored on any suitable computer-readable media such as RAMs, ROMs, flash memory, EEPROMs, optical devices (CD or DVD), hard drives, floppy drives, memory sticks (e.g., SD cards, USB flash drives), cloud-based services (e.g., cloud storage), magnetic storage devices, Solid-State Drives (SSDs), or any suitable device. The computer-executable component is preferably a general or application-specific processor, but any suitable dedicated hardware or hardware/firmware combination device can alternatively or additionally execute the instructions.

The systems and methods of the preferred embodiments may additionally, or alternatively, be implemented on an integrated data analytics software application and/or software architecture such as those offered by SAS Institute Inc. of Cary, N.C., USA. Merely for illustration, the systems and methods of the preferred embodiments may be implemented using or integrated with one or more SAS software tools such as SAS® Viya™ which is developed and provided by SAS Institute Inc. of Cary, N.C., USA.

Although omitted for conciseness, the preferred embodiments include every combination and permutation of the implementations of the systems and methods described herein.

As a person skilled in the art will recognize from the previous detailed description and from the figures and claims, modifications and changes can be made to the embodiments of the disclosure without departing from the scope of the various described embodiments.

What is claimed is:

1. A computer-program product comprising a non-transitory machine-readable storage medium storing computer instructions that, when executed by one or more processors, perform operations comprising:

52

creating a geospatial dataset comprising features for a plurality of geographical cells within a target spatial area;

executing, via a flood simulation model, a computer simulation that simulates flooding within the target spatial area based on an input of (1) the geospatial dataset and (2) rainfall intensity data of one or more flooding events associated with the target spatial area, wherein executing the computer simulation includes: simulating inundation levels of the plurality of geographical cells during the one or more flooding events, and

simulating a flow of water across the plurality of geographical cells during the one or more flooding events;

converting the inundation levels of the plurality of geographical cells to normalized inundation scores for the plurality of geographical cells;

detecting one or more clusters of interconnected geographical cells based on the simulating of the flow of water;

determining, via an optimization algorithm, a set of optimal sensor locations in the target spatial area based on the normalized inundation scores and the one or more clusters of interconnected geographical cells;

generating a sensor location map that includes the plurality of geographical cells and a set of geospatial markers identifying the set of optimal sensor locations;

generating a visual output that displays the sensor location map after generating the sensor location map; and positioning one or more sensors at one or more of the optimal sensor locations based at least in part on the visual output that displays the sensor location map.

2. The computer-program product according to claim 1, wherein simulating the inundation levels of the plurality of geographical cells during a respective flooding event includes:

identifying, from the rainfall intensity data, precipitation amounts observed during a plurality of timesteps of the respective flooding event,

simulating, via the flood simulation model, the precipitation amounts observed during the plurality of timesteps on the plurality of geographical cells, and

determining maximum inundation levels of the plurality of geographical cells based at least on the simulating of the precipitation amounts.

3. The computer-program product according to claim 2, wherein determining a maximum inundation level of a respective geographical cell includes:

identifying an inundation level of the respective geographical cell at a first timestep of the respective flooding event,

identifying inundation levels of the respective geographical cell at a plurality of other timesteps of the respective flooding event, and

using the inundation level identified at the first timestep as the maximum inundation level if the inundation level identified at the first timestep is greater than the inundation levels identified at the plurality of other timesteps.

4. The computer-program product according to claim 1, wherein:

the inundation levels of the plurality of geographical cells include maximum inundation levels of the plurality of geographical cells during a respective flooding event, and

53

converting an inundation level of a respective geographical cell to a normalized inundation score for the respective geographical cell includes:

identifying an overall maximum inundation level among the maximum inundation levels, and computing the normalized inundation score by dividing the inundation level of the respective geographical cell by the overall maximum inundation level.

5. The computer-program product according to claim 1, wherein simulating an inundation level of a respective geographical cell includes:

identifying precipitation amounts observed during a plurality of timesteps of a respective flooding event, simulating the precipitation amounts observed during the plurality of timesteps on the respective geographical cell, computing inundation levels of the respective geographical cell across the plurality of timesteps, and using a maximum inundation level of the inundation levels computed across the plurality of timesteps as the inundation level of the respective geographical cell.

6. The computer-program product according to claim 1, wherein computing an inundation level of a respective geographical cell at a respective timestep includes:

- (a) identifying a precipitation amount observed during the respective timestep,
- (b) identifying an amount of water flowing from other geographical cells into the respective geographical cell during the respective timestep,
- (c) identifying an amount of water flowing from the respective geographical cell to the other geographical cells during the respective timestep,
- (d) identifying an amount of water lost due to infiltration during the respective timestep,
- (e) identifying an amount of water removed via storm drain outflow during the respective timestep, and
- (f) computing the inundation level of the respective geographical cell using (a)-(e).

7. The computer-program product according to claim 6, wherein computing the inundation level of the respective geographical cell using (a)-(e) includes:

summing the precipitation amount identified in (a) with the amount of water identified in (b), and subtracting the amount of water identified in (c), the amount of water identified in (d), and the amount of water identified in (e) from a result of the summing.

8. The computer-program product according to claim 1, wherein:

the flow of water across the plurality of geographical cells is represented using a graph data structure, and the graph data structure includes:

a plurality of nodes corresponding to the plurality of geographical cells, and
a plurality of edges specifying water flow trajectories among the plurality of nodes.

9. The computer-program product according to claim 8, wherein a respective edge extending from a first respective node to a second respective node specifies a water flow trajectory that indicates water moves from a geographical cell corresponding to the first respective node to a geographical cell corresponding to the second respective node during flooding.

10. The computer-program product according to claim 1, wherein:

the flow of water across the plurality of geographical cells is represented using a graph data structure, and

54

detecting the one or more clusters of interconnected geographical cells includes:

identifying that a first plurality of nodes of the graph data structure are connected via a first plurality of edges; and

detecting the first plurality of nodes as a first cluster of interconnected geographical cells based on identifying that the first plurality of nodes are connected via the first plurality of edges.

11. The computer-program product according to claim 10, wherein:

detecting the one or more clusters of interconnected geographical cells further includes:

identifying that a second plurality of nodes of the graph data structure are connected via a second plurality of edges;

adding the second plurality of nodes to the first cluster of interconnected geographical cells if there is at least one edge connecting a node from the first plurality of nodes to a node of the second plurality of nodes, and

detecting the second plurality of nodes as a second cluster of interconnected geographical cells if there is not at least one edge connecting the node from the first plurality of nodes to the node of the second plurality of nodes.

12. The computer-program product according to claim 1, wherein a respective cluster of interconnected geographical cells:

corresponds to a respective sub-catchment within the target spatial area,

indicates flow characteristics of the respective sub-catchment, and

indicates a retention size of the respective sub-catchment.

13. The computer-program product according to claim 1, wherein:

the optimization algorithm maximizes a multi-criterion objective function to determine the set of optimal sensor locations, and

the multi-criterion objective function is a weighted sum of:

an inundation coverage objective function that prioritizes sensor locations in geographical cells that yield a highest aggregated normalized inundation score across different combinations of the plurality of geographical cells, and

a cluster coverage objective function that prioritizes sensor locations in geographical cells that maximize coverage across the one or more clusters of interconnected geographical cells.

14. The computer-program product according to claim 13, wherein the inundation coverage objective function prioritizes the sensor locations in the geographical cells that:

are assigned a criticality multiplier value due a presence of a pre-defined type of asset, and

yield the highest aggregated normalized inundation score across different combinations of the plurality of geographical cells.

15. The computer-program product according to claim 1, wherein the optimization algorithm includes:

a constraint that constrains the set of optimal sensor locations to a pre-defined number of optimal sensor locations, and

a constraint that constrains the set of optimal sensor locations to include one or more pre-determined critical locations.

55

16. The computer-program product according to claim 1, wherein creating the geospatial dataset includes:
 obtaining a representation of the target spatial area,
 dividing the representation of the target spatial area into
 the plurality of geographical cells,
 obtaining a plurality of feature maps associated with the
 target spatial area,
 extracting the features of the plurality of geographical
 cells from the plurality of feature maps, and
 adding entries to the geospatial dataset that correspond to
 the plurality of geographical cells and that store the
 features extracted for the plurality of geographical
 cells.

17. The computer-program product according to claim 1,
 wherein:
 the features for the plurality of geographical cells includes
 features for a respective geographical cell of the
 plurality of geographical cells, and
 the features for the respective geographical cell include:
 one or more identification features associated with the
 respective geographical cell,
 one or more topographical features associated with the
 respective geographical cell, and
 one or more soil and drainage features associated with
 the respective geographical cell.

18. The computer-program product according to claim 17,
 wherein the one or more identification features associated
 with the respective geographical cell include an identifica-
 tion value associated with the respective geographical cell.

19. The computer-program product according to claim 17,
 wherein the one or more topographical features associated
 with the respective geographical cell include:
 an elevation of the respective geographical cell,
 a percentage of impervious surface within the respective
 geographical cell, and
 a land cover classification associated with the respective
 geographical cell.

20. The computer-program product according to claim 17,
 wherein the one or more soil and drainage features associated
 with the respective geographical cell include:
 a soil type associated with the respective geographical
 cell,
 a type of storm drain present in the respective geographi-
 cal cell, and
 a maximum capacity of the storm drain in the respective
 geographical cell.

21. The computer-program program product according to
 claim 1, wherein:
 generating the sensor location map includes overlaying
 the set of geospatial markers on the plurality of geo-
 graphical cells; and
 generating the visual output comprises displaying the
 sensor location map in a graphical user interface after
 generating the sensor location map.

22. The computer-program product according to claim 1,
 wherein:

generating the visual output comprises generating a report
 comprising a visual depiction of the sensor location
 map after generating the sensor location map.

23. A computer-implemented method, comprising:
 creating a geospatial dataset comprising features for a
 plurality of geographical cells within a target spatial
 area;
 executing, via a flood simulation model, a computer
 simulation that simulates flooding within the target
 spatial area based on an input of (1) the geospatial
 dataset and (2) rainfall intensity data of one or more

56

flooding events associated with the target spatial area,
 wherein executing the computer simulation includes:
 simulating inundation levels of the plurality of geo-
 graphical cells during the one or more flooding
 events, and
 simulating a flow of water across the plurality of
 geographical cells during the one or more flooding
 events;
 converting the inundation levels of the plurality of geo-
 graphical cells to normalized inundation scores for the
 plurality of geographical cells;
 detecting one or more clusters of interconnected geo-
 graphical cells based on the simulating of the flow of
 water;
 determining, via an optimization algorithm, a set of
 optimal sensor locations in the target spatial area based
 on the normalized inundation scores and the one or
 more clusters of interconnected geographical cells;
 generating a sensor location map that includes the plu-
 rality of geographical cells and a set of geospatial
 markers identifying the set of optimal sensor locations;
 generating a visual output that displays the sensor location
 map after generating the sensor location map; and
 positioning one or more sensors at one or more of the
 optimal sensor locations based at least in part on the
 visual output that displays the sensor location map.

24. The computer-implemented method according to
 claim 23, wherein simulating the inundation levels of the
 plurality of geographical cells during a respective flooding
 event includes:

identifying, from the rainfall intensity data, precipitation
 amounts observed during a plurality of timesteps of the
 respective flooding event,
 simulating, via the flood simulation model, the precipita-
 tion amounts observed during the plurality of timesteps
 on the plurality of geographical cells, and
 determining maximum inundation levels of the plurality
 of geographical cells based at least on the simulating of
 the precipitation amounts.

25. The computer-implemented method according to
 claim 24, wherein determining a maximum inundation level
 of a respective geographical cell includes:

identifying an inundation level of the respective geo-
 graphical cell at a first timestep of the respective
 flooding event,
 identifying inundation levels of the respective geographi-
 cal cell at a plurality of other timesteps of the respective
 flooding event, and
 using the inundation level identified at the first timestep as
 the maximum inundation level if the inundation level
 identified at the first timestep is greater than the inun-
 dation levels identified at the plurality of other
 timesteps.

26. The computer-implemented method according to
 claim 23, wherein:

the inundation levels of the plurality of geographical cells
 include maximum inundation levels of the plurality of
 geographical cells during a respective flooding event,
 and

converting an inundation level of a respective geographi-
 cal cell to a normalized inundation score for the respec-
 tive geographical cell includes:
 identifying an overall maximum inundation level
 among the maximum inundation levels, and
 computing the normalized inundation score by dividing
 the inundation level of the respective geographical
 cell by the overall maximum inundation level.

57

27. The computer-implemented method according to claim **23**, wherein simulating an inundation level of a respective geographical cell includes:

identifying precipitation amounts observed during a plurality of timesteps of a respective flooding event,
simulating the precipitation amounts observed during the plurality of timesteps on the respective geographical cell,
computing inundation levels of the respective geographical cell across the plurality of timesteps, and
using a maximum inundation level of the inundation levels computed across the plurality of timesteps as the inundation level of the respective geographical cell.

28. The computer-implemented method according to claim **23**, wherein computing an inundation level of a respective geographical cell at a respective timestep includes:

- (a) identifying a precipitation amount observed during the respective timestep,
- (b) identifying an amount of water flowing from other geographical cells into the respective geographical cell during the respective timestep,
- (c) identifying an amount of water flowing from the respective geographical cell to the other geographical cells during the respective timestep,
- (d) identifying an amount of water lost due to infiltration during the respective timestep,
- (e) identifying an amount of water removed via storm drain outflow during the respective timestep, and
- (f) computing the inundation level of the respective geographical cell using (a)-(e).

29. A computer-implemented system, comprising:

one or more processors;
a memory;
a computer-readable medium operably coupled to the one or more processors, the computer-readable medium having computer-readable instructions stored thereon that, when executed by the one or more processors, cause a computing device to perform operations comprising:
creating a geospatial dataset comprising features for a plurality of geographical cells within a target spatial area;

5

10

15

20

25

30

35

40

58

executing, via a flood simulation model, a computer simulation that simulates flooding within the target spatial area based on an input of (1) the geospatial dataset and (2) rainfall intensity data of one or more flooding events associated with the target spatial area, wherein executing the computer simulation includes: simulating inundation levels of the plurality of geographical cells during the one or more flooding events, and

simulating a flow of water across the plurality of geographical cells during the one or more flooding events;

converting the inundation levels of the plurality of geographical cells to normalized inundation scores for the plurality of geographical cells;

detecting one or more clusters of interconnected geographical cells based on the simulating of the flow of water;

determining, via an optimization algorithm, a set of optimal sensor locations in the target spatial area based on the normalized inundation scores and the one or more clusters of interconnected geographical cells; generating a sensor location map that includes the plurality of geographical cells and a set of geospatial markers identifying the set of optimal sensor locations; generating a visual output that displays the sensor location map after generating the sensor location map; and positioning one or more sensors at one or more of the optimal sensor locations based at least in part on the visual output that displays the sensor location map.

30. The computer-implemented system according to claim **29**, wherein simulating the inundation levels of the plurality of geographical cells during a respective flooding event includes:

identifying, from the rainfall intensity data, precipitation amounts observed during a plurality of timesteps of the respective flooding event,

simulating, via the flood simulation model, the precipitation amounts observed during the plurality of timesteps on the plurality of geographical cells, and

determining maximum inundation levels of the plurality of geographical cells based at least on the simulating of the precipitation amounts.

* * * * *