US 20250267068A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2025/0267068 A1**

Nguyen (43) **Pub. Date: Aug. 21, 2025**

(54) **SYSTEM FOR CLOUD SOLUTION MIGRATION AND MANAGEMENT**

(71) Applicant: **T-Mobile USA, Inc.**, Bellevue, WA (US)

(72) Inventor: **Phi Hoang Nguyen**, Lacey, WA (US)

(21) Appl. No.: **18/443,914**

(22) Filed: **Feb. 16, 2024**

Publication Classification

(51) **Int. Cl.**

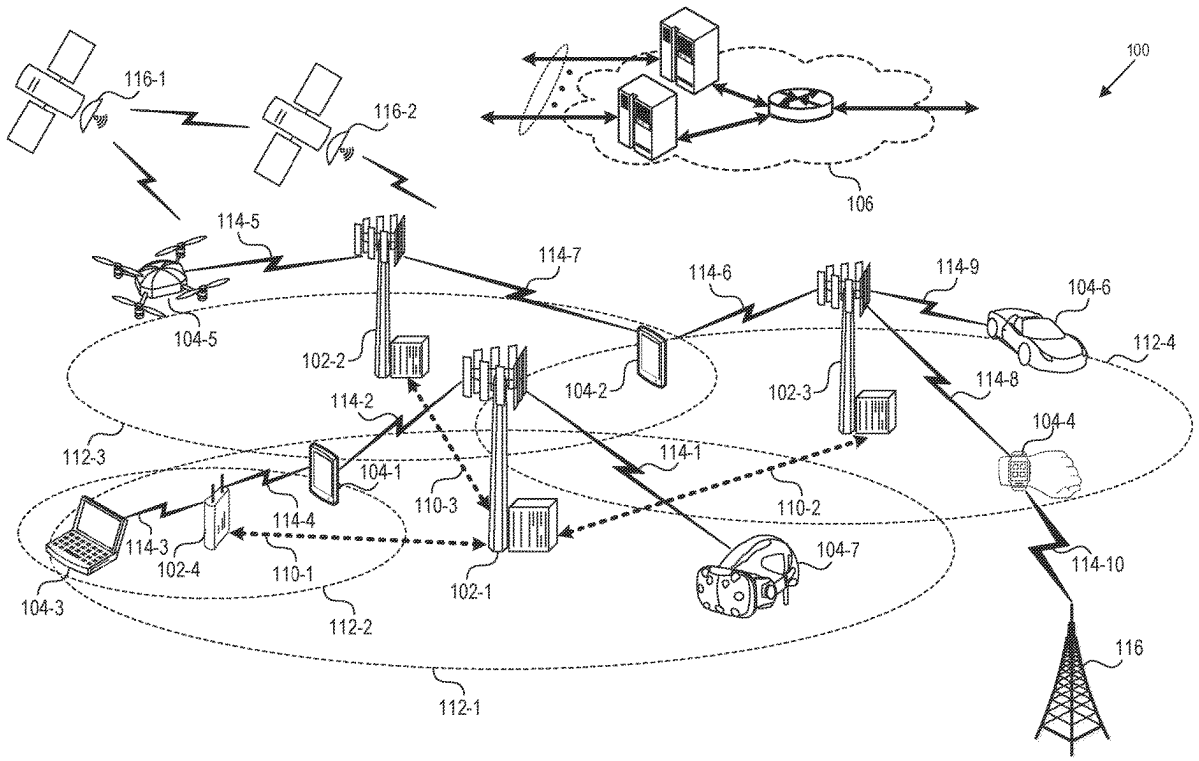
H04L 41/0895	(2022.01)
G06F 8/30	(2018.01)
G06F 9/48	(2006.01)
G06F 9/54	(2006.01)
H04L 41/0806	(2022.01)

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

CPC ***H04L 41/0895*** (2022.05); ***G06F 9/4875*** (2013.01); ***G06F 9/547*** (2013.01); ***H04L 41/0806*** (2013.01); *G06F 8/31* (2013.01)

(57) **ABSTRACT**

The system can receive data, including a specification for a cloud solution, indicating a type of cloud solution and at least one requirement. The system can determine a list of cloud service providers with cloud infrastructure capable of hosting the cloud solution and an ability to satisfy the at least one requirement. The system can rank the cloud service providers based on the capability to host the cloud solution and the ability to satisfy the at least one requirement. The system can select the cloud service provider based on the ranking. The system can generate an implementation procedure based on an Application Programming Interface (API) specification, where the API specification determines a requirement for the cloud solution to be deployed on the cloud infrastructure. The system can deploy, using a large language model (LLM), the cloud solution on the cloud infrastructure in accordance with the implementation procedure.



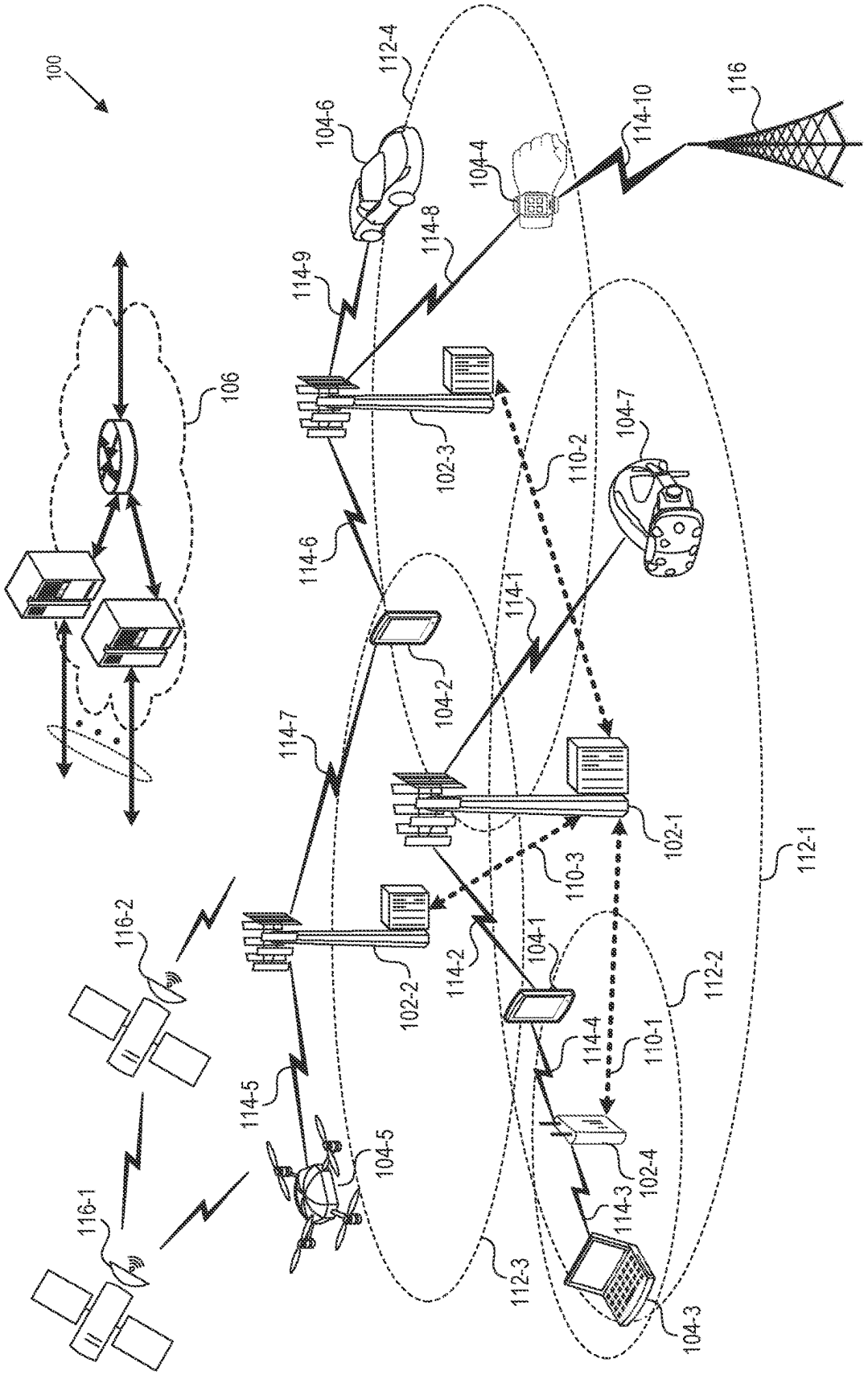

100
106
112-4
104-6
104-4
114-10
116
114-9
114-8
102-3
110-2
104-7
114-6
114-1
104-2
114-7
102-1
110-3
112-1
116-2
102-2
114-2
104-1
112-2
114-4
110-1
114-5
104-5
102-4
114-3
116-1
112-3
104-3

FIG. 1

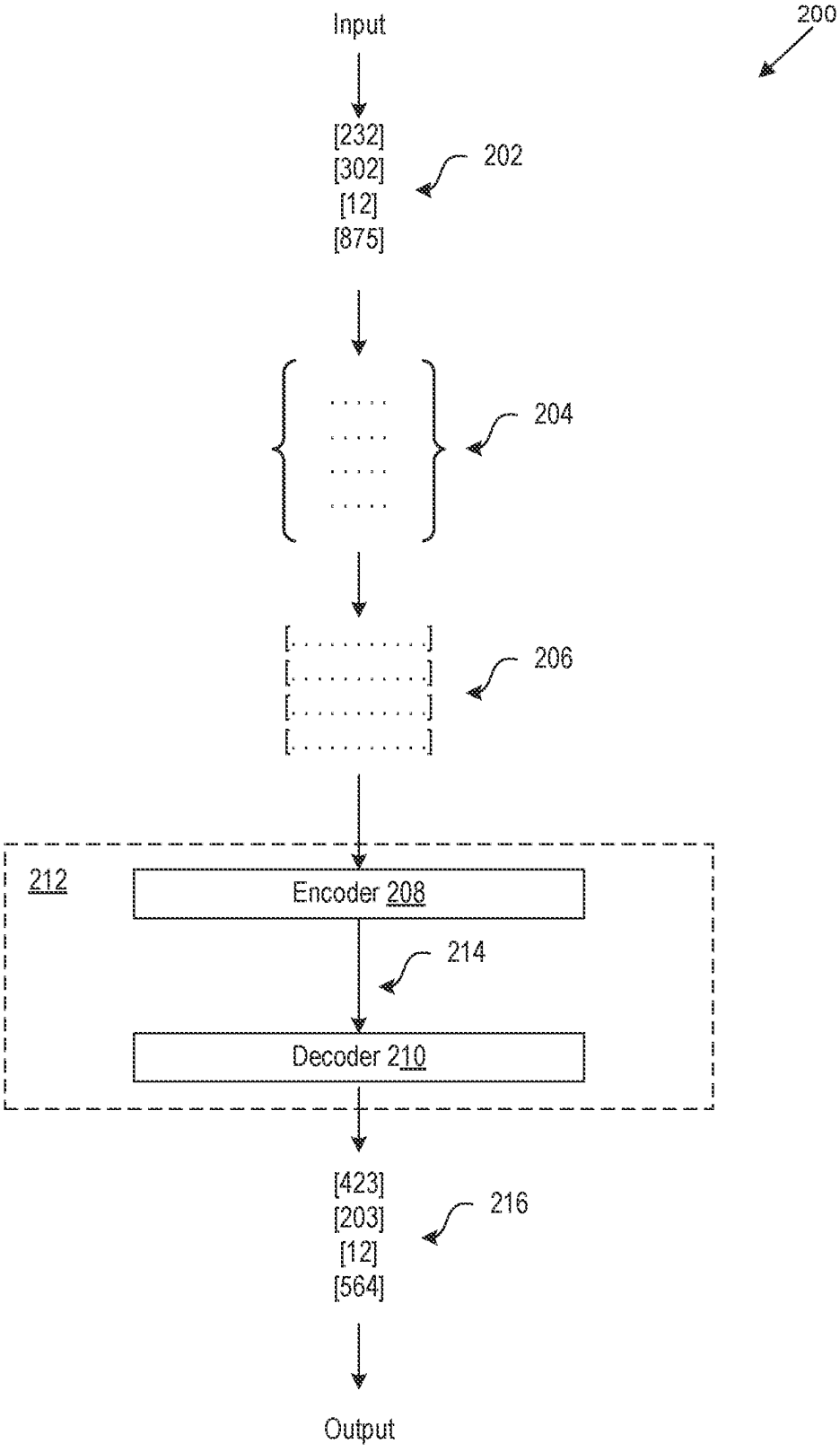

Input
200
[232]
[302]
[12]
[875]
202
204
206
212
Encoder 208
214
Decoder 210
[423]
[203]
[12]
[564]
216
Output

FIG. 2

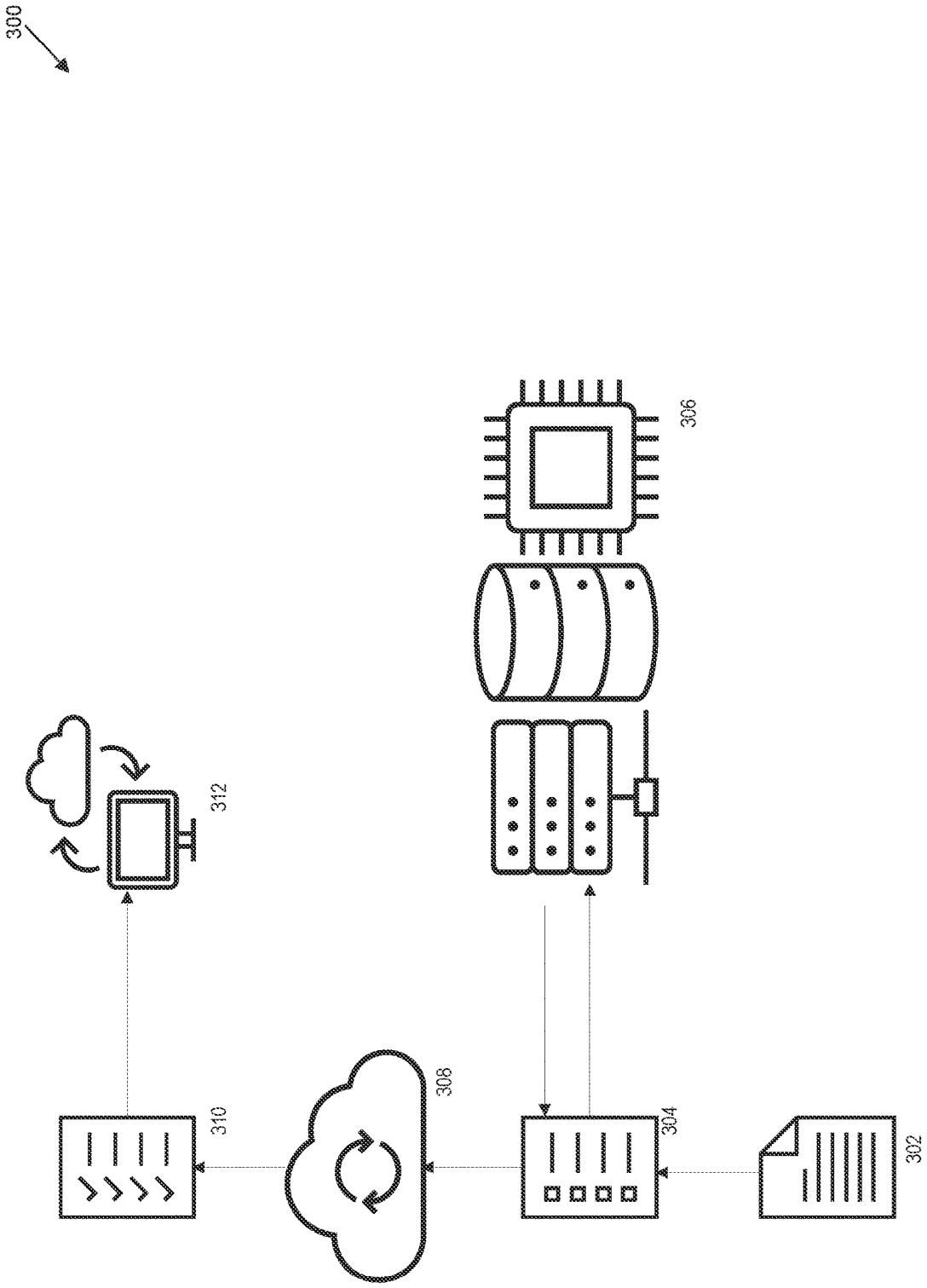


FIG. 3

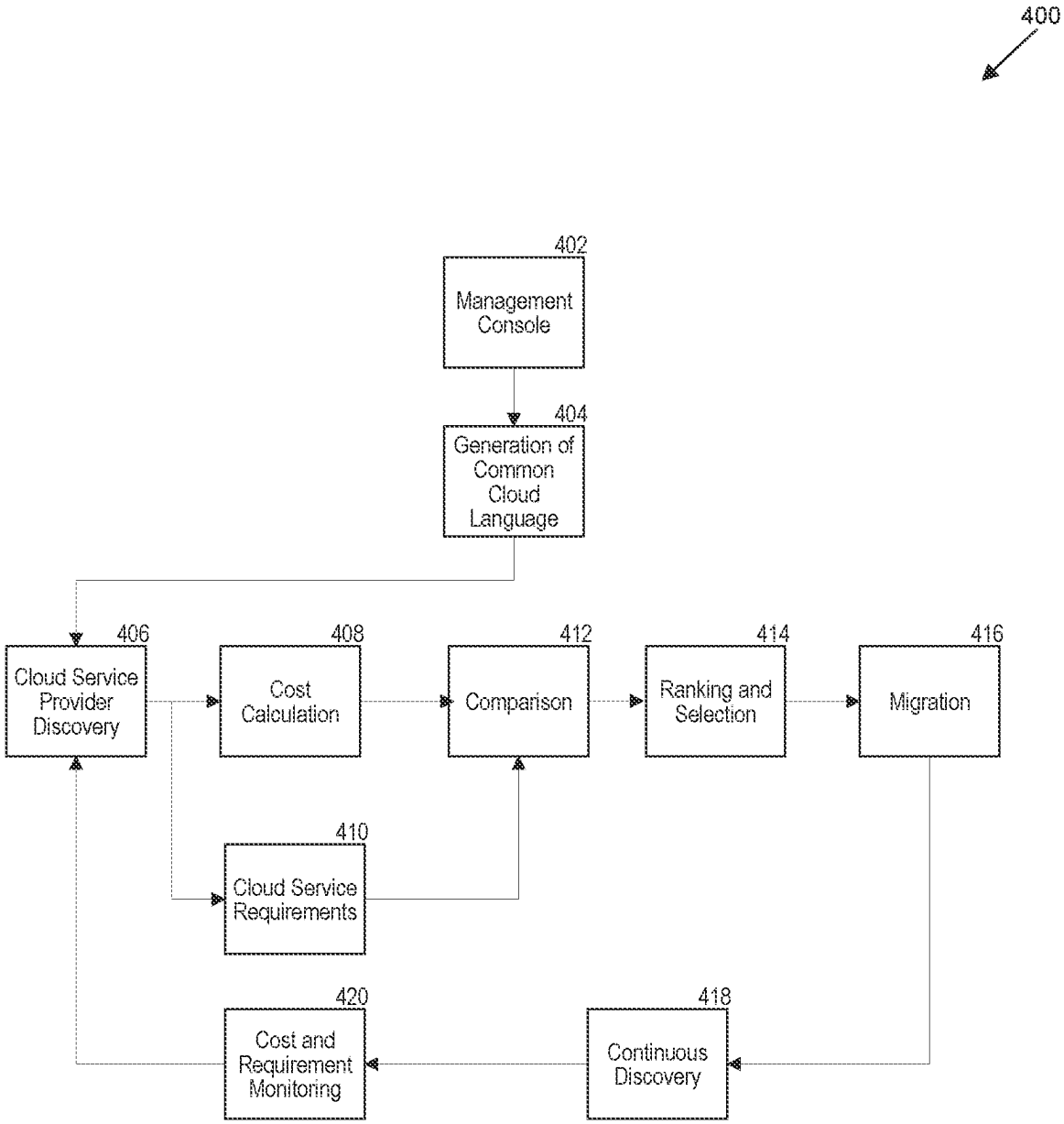

400
402
Management Console
404
Generation of Common Cloud Language
406
Cloud Service Provider Discovery
408
Cost Calculation
412
Comparison
414
Ranking and Selection
416
Migration
410
Cloud Service Requirements
420
Cost and Requirement Monitoring
418
Continuous Discovery

FIG. 4

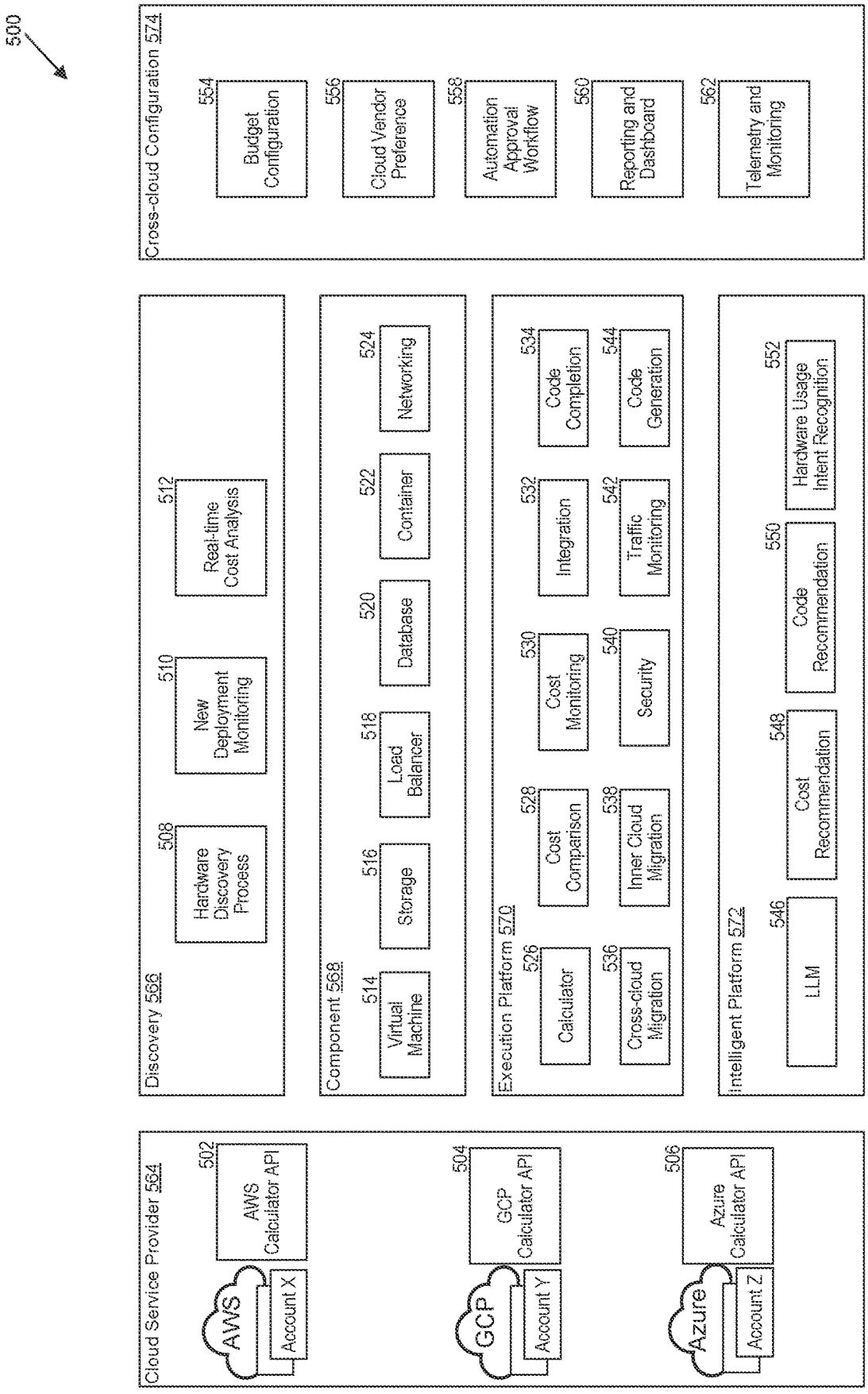

500
Cross-cloud Configuration 574
554
Budget Configuration
556
Cloud Vendor Preference
558
Automation Approval Workflow
560
Reporting and Dashboard
562
Telemetry and Monitoring
Discovery 566
508
Hardware Discovery Process
510
New Deployment Monitoring
512
Real-time Cost Analysis
Component 568
514
Virtual Machine
516
Storage
518
Load Balancer
520
Database
522
Container
524
Networking
Execution Platform 570
526
Calculator
528
Cost Comparison
530
Cost Monitoring
532
Integration
534
Code Completion
536
Cross-cloud Migration
538
Inner Cloud Migration
540
Security
542
Traffic Monitoring
544
Code Generation
Intelligent Platform 572
546
LLM
548
Cost Recommendation
550
Code Recommendation
552
Hardware Usage Intent Recognition
Cloud Service Provider 564
502
AWS Calculator API
AWS
Account X
504
GCP Calculator API
GCP
Account Y
506
Azure Calculator API
Azure
Account Z

FIG. 5

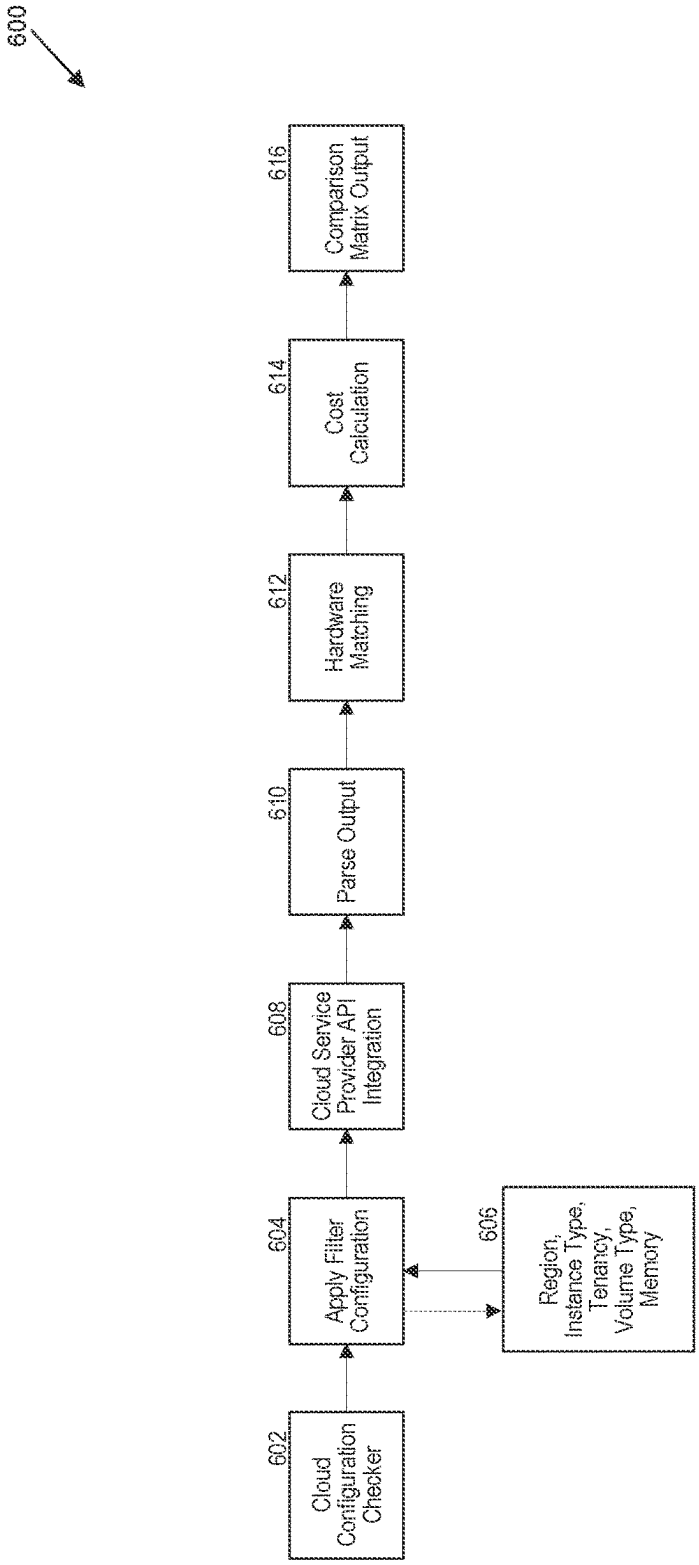

600
602
Cloud Configuration Checker
604
Apply Filter Configuration
606
Region, Instance Type, Tenancy, Volume Type, Memory
608
Cloud Service Provider API Integration
610
Parse Output
612
Hardware Matching
614
Cost Calculation
616
Comparison Matrix Output

FIG. 6

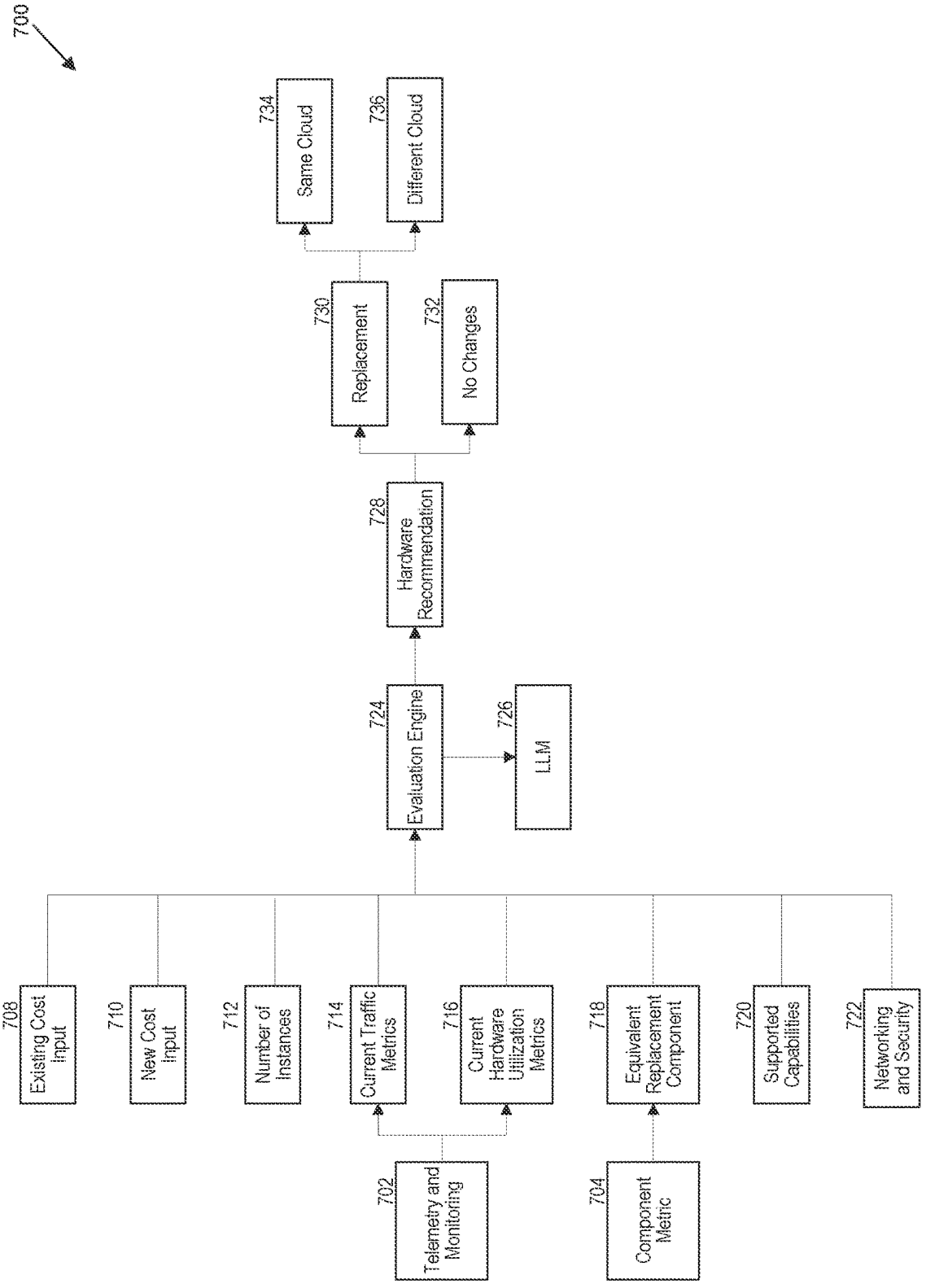

700
734
Same Cloud
736
Different Cloud
730
Replacement
732
No Changes
728
Hardware Recommendation
724
Evaluation Engine
726
LLM
708
Existing Cost Input
710
New Cost Input
712
Number of Instances
714
Current Traffic Metrics
716
Current Hardware Utilization Metrics
718
Equivalent Replacement Component
720
Supported Capabilities
722
Networking and Security
702
Telemetry and Monitoring
704
Component Metric

FIG. 7

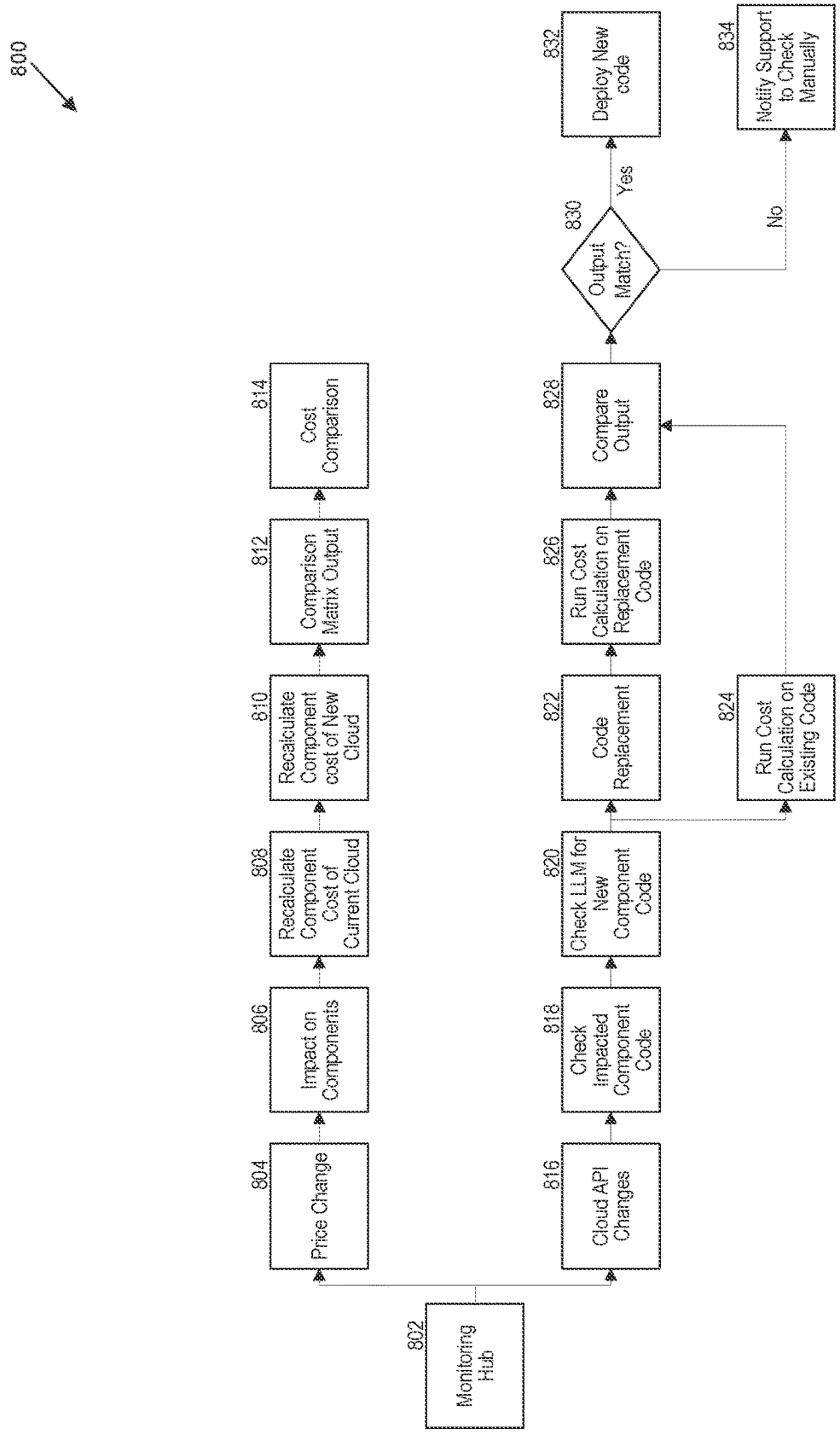

800
802
Monitoring Hub
804
Price Change
806
Impact on Components
808
Recalculate Component Cost of Current Cloud
810
Recalculate Component cost of New Cloud
812
Comparison Matrix Output
814
Cost Comparison
816
Cloud API Changes
818
Check Impacted Component Code
820
Check LLM for New Component Code
822
Code Replacement
824
Run Cost Calculation on Existing Code
826
Run Cost Calculation on Replacement Code
828
Compare Output
830
Output Match?
Yes
No
832
Deploy New code
834
Notify Support to Check Manually

FIG. 8

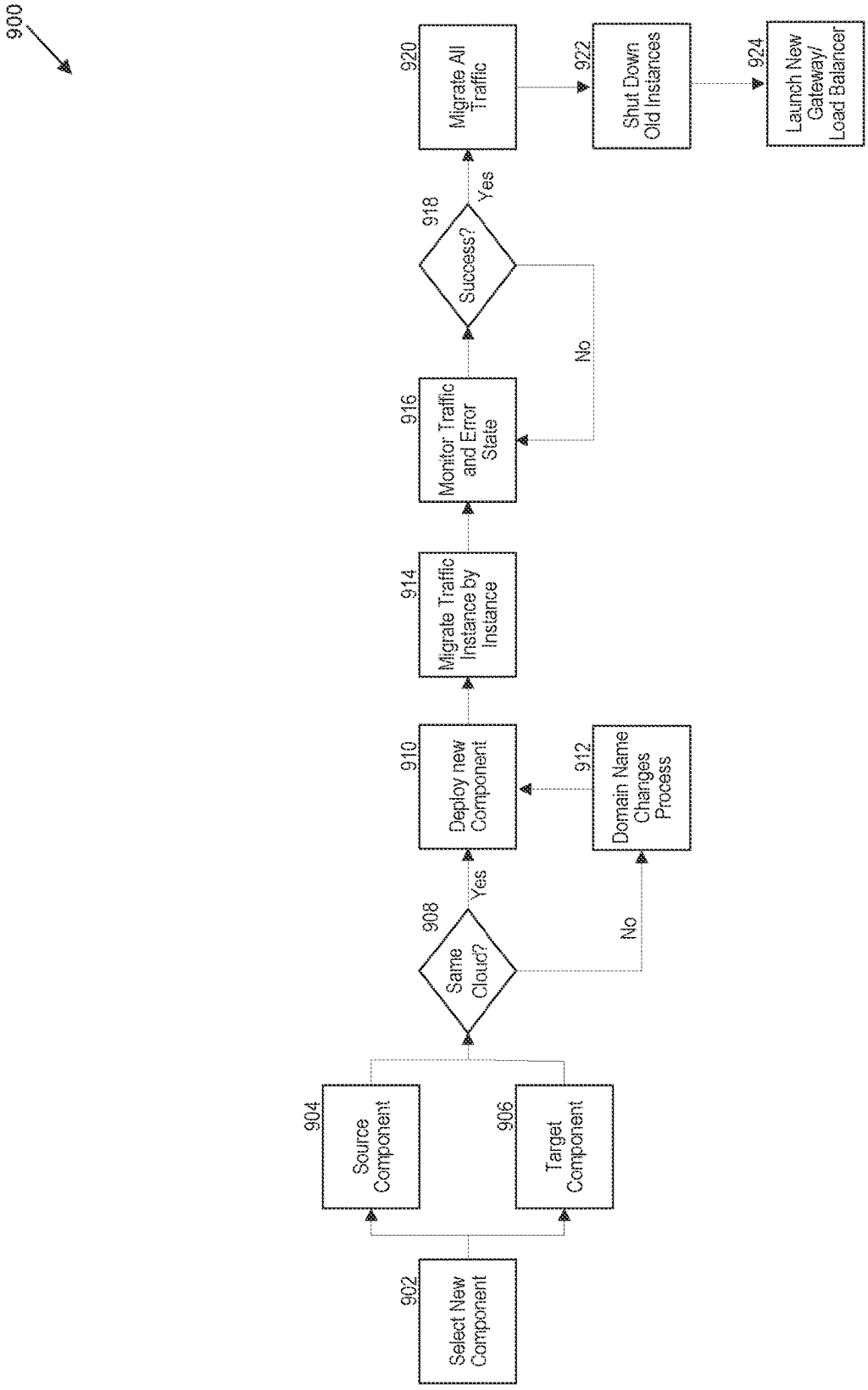

900
902
Select New Component
904
Source Component
906
Target Component
908
Same Cloud?
Yes
No
910
Deploy new Component
912
Domain Name Changes Process
914
Migrate Traffic Instance by Instance
916
Monitor Traffic and Error State
918
Success?
Yes
No
920
Migrate All Traffic
922
Shut Down Old Instances
924
Launch New Gateway/ Load Balancer

FIG. 9

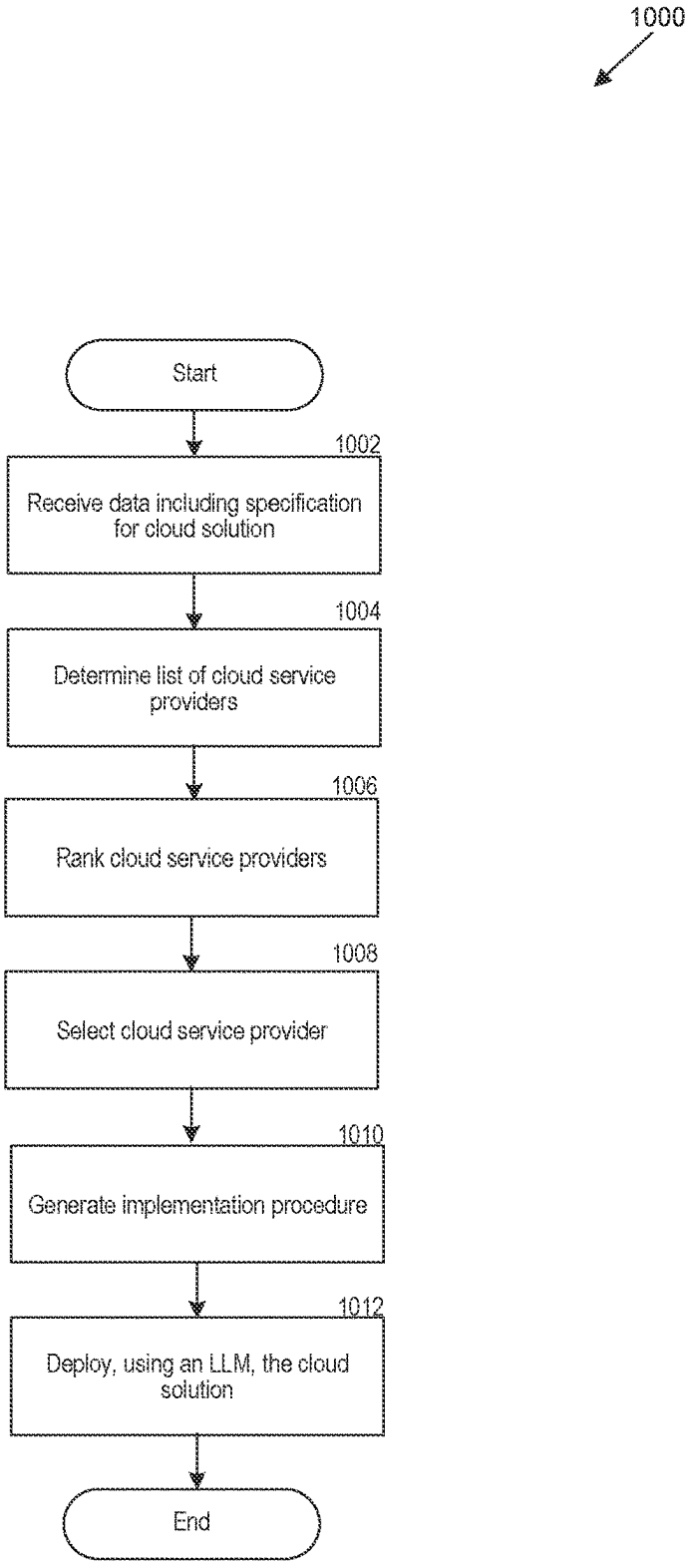

1000
Start
1002
Receive data including specification for cloud solution
1004
Determine list of cloud service providers
1006
Rank cloud service providers
1008
Select cloud service provider
1010
Generate implementation procedure
1012
Deploy, using an LLM, the cloud solution
End

FIG. 10

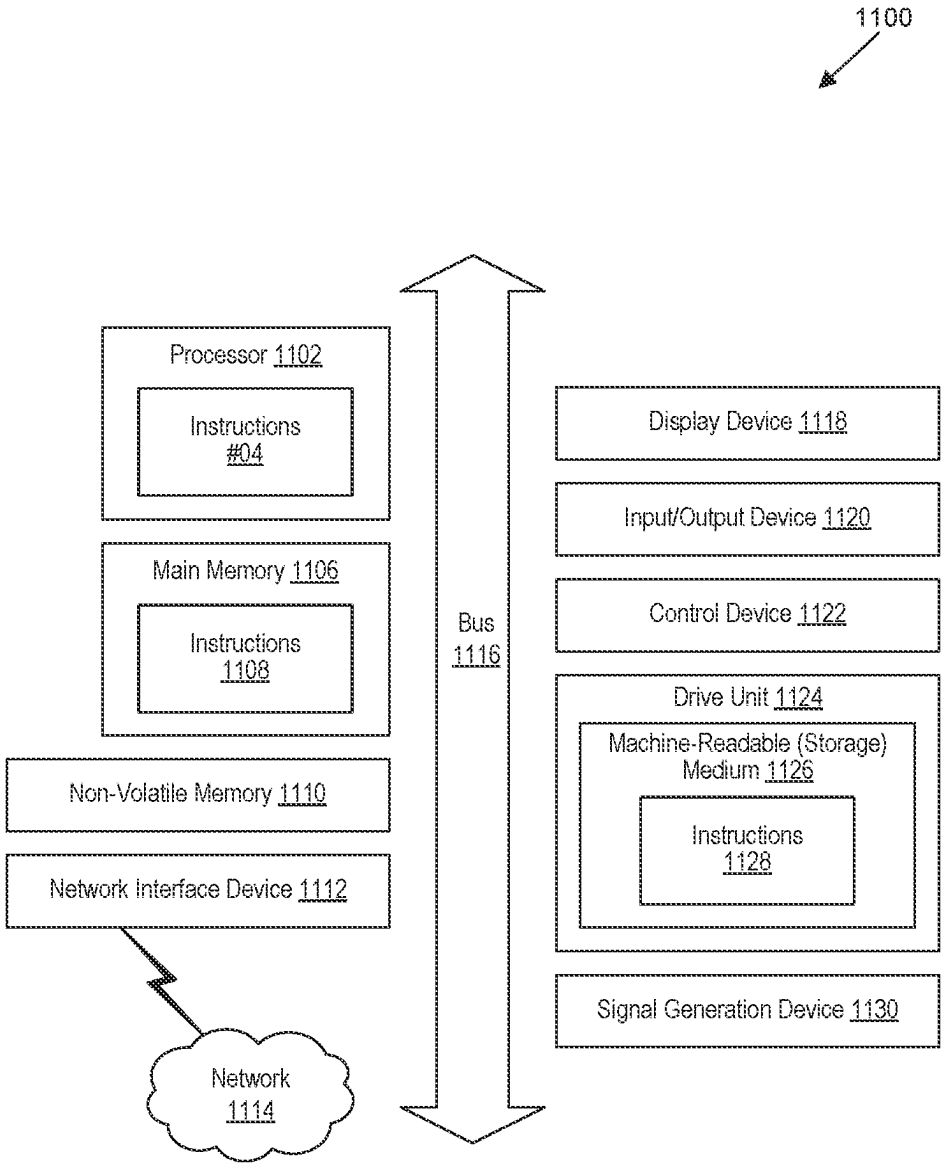

1100
Processor 1102
Instructions #04
Main Memory 1106
Instructions 1108
Non-Volatile Memory 1110
Network Interface Device 1112
Network 1114
Bus 1116
Display Device 1118
Input/Output Device 1120
Control Device 1122
Drive Unit 1124
Machine-Readable (Storage) Medium 1126
Instructions 1128
Signal Generation Device 1130

FIG. 11

SYSTEM FOR CLOUD SOLUTION MIGRATION AND MANAGEMENT

BACKGROUND

[0001] Cloud computing is the on-demand availability of computer system resources, especially data storage (cloud storage) and computing power, without direct active management by the user. Large clouds often have functions distributed over multiple locations, each of which is a data center. Cloud computing relies on sharing of resources to achieve coherence and typically uses a pay-as-you-go model, which can help in reducing capital expenses but may also lead to unexpected operating expenses for users.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] Detailed descriptions of implementations of the present invention will be described and explained through the use of the accompanying drawings.

[0003] FIG. **1** is a block diagram that illustrates a wireless communications system that can implement aspects of the present technology.

[0004] FIG. **2** is a block diagram that illustrates 5G core network functions (NFs) that can implement aspects of the present technology.

[0005] FIG. **3** illustrates an embodiment for selecting and implementing a new cloud solution.

[0006] FIG. **4** illustrates a flow diagram for selecting a new cloud provider for an existing cloud solution.

[0007] FIG. **5** illustrates a block diagram of various features of an embodiment of the system.

[0008] FIG. **6** illustrates a flow diagram of an embodiment of a calculator tool.

[0009] FIG. **7** illustrates a flow diagram for an embodiment of the recommendation process.

[0010] FIG. **8** illustrates a flow diagram of an embodiment of a monitoring hub.

[0011] FIG. **9** illustrates a flow diagram of an embodiment for replacing a cloud infrastructure component.

[0012] FIG. **10** is a flow diagram illustrating an embodiment of a cloud migration process.

[0013] FIG. **11** is a block diagram that illustrates an example of a computer system in which at least some operations described herein can be implemented.

[0014] The technologies described herein will become more apparent to those skilled in the art from studying the Detailed Description in conjunction with the drawings. Embodiments or implementations describing aspects of the invention are illustrated by way of example, and the same references can indicate similar elements. While the drawings depict various implementations for the purpose of illustration, those skilled in the art will recognize that alternative implementations can be employed without departing from the principles of the present technologies. Accordingly, while specific implementations are shown in the drawings, the technology is amenable to various modifications.

DETAILED DESCRIPTION

[0015] The disclosed technology relates to a system that can seamlessly implement a new cloud solution on the cloud infrastructure of a cloud service provider. In one example, the system receives a specification for a cloud solution. The specification defines a series of parameters for the cloud solution, such as the type of cloud solution and the requirements for the cloud solution. A cloud solution includes a cloud service provider, which refers to an entity that offers on-demand, scalable computing resources over the Internet used to implement a cloud solution. A cloud solution is a computing service hosted on the cloud infrastructure of the cloud service provider over the Internet. A cloud solution can be an infrastructure-as-a-service (IaaS), platform-as-a-service (PaaS), software-as-a-service (SaaS), or function-as-a-service (FaaS). IaaS is a form of cloud computing service that offers on-demand essential computing, storage, and networking resources. PaaS is a complete development and deployment environment in the cloud, with resources that enable you to deliver everything from simple cloud-based applications to sophisticated, cloud-enabled enterprise applications. SaaS allows users to connect to and use cloud-based applications over the Internet-for example, email, calendaring, and productivity tools. FaaS is an event-driven execution model that allows developers to build, run, and manage application packages as functions without maintaining any infrastructure. For example, cloud infrastructure can refer to the hardware components and software used to host the cloud solution.

[0016] In one example, the requirements for a cloud system include the ability of the cloud infrastructure to support a threshold amount of user traffic volume or a certain user traffic type. In another example, the requirements for the cloud system can be a provisioning speed or reliability of the cloud infrastructure, the scalability of the cloud solution on the cloud infrastructure, and/or the security protocols, technology, or controls enacted by the cloud service provider. The scalability of the cloud infrastructure refers to the ability to increase or decrease the amount of computing power, storage, or bandwidth resources delivered to the cloud solution.

[0017] The disclosed technology solves issues resulting from different cloud service providers that enact different criteria to deploy a cloud solution, creating difficulties in the initial deployment of the cloud solution and in migrating the cloud solution to a new cloud service provider. As used herein, "cloud migration" refers to the process of moving data, applications, or workloads from one cloud environment to another or from on-premises systems to the cloud. Cloud migration can be complex, time consuming, and costly, especially when there are incompatibility issues between different cloud platforms or architectures. Cloud migration can also cause downtime, performance degradation, or data loss if not planned and executed properly. The disclosed technology relates to a system for implementing a cloud solution on the cloud infrastructure of a cloud service provider.

[0018] Receiving the specification for the cloud solution allows the system to determine the cloud infrastructure requirements for the cloud solution. For example, the system can receive the specification over a 5G network or any other wireless network. The system can determine a list of cloud service providers with cloud infrastructure capable of hosting the cloud system and an ability to satisfy the requirements of the cloud solution. The system can rank the cloud service providers based on the capability of the cloud service provider to host the cloud solution and the ability of the cloud service provider to satisfy the requirements of the cloud system. The system can select a cloud service provider based on the ranking. The system can determine an implementation procedure. The implementation procedure is a

process of deploying the cloud solution on the cloud infrastructure. The system can deploy the cloud solution using the implementation procedure.

[0019] In another embodiment, the implementation procedure causes the system to generate a common cloud source code for the cloud system. The system can use a large language model (LLM) to generate the common cloud source code. The common cloud source code is a series of instructions that define the cloud solution. The common cloud source code is a generic programming syntax that can be converted to different programming languages or syntaxes used by the cloud service providers. Implementing a cloud solution onto the cloud infrastructure entails using a unique programming language and syntax chosen by the cloud service provider. The system can convert, using the LLM, the common cloud source code to the programming syntax of the selected cloud service provider. The system can deploy the cloud solution on the cloud infrastructure using the converted programming syntax.

[0020] The description and associated drawings are illustrative examples and are not to be construed as limiting. This disclosure provides certain details for a thorough understanding and enabling description of these examples. One skilled in the relevant technology will understand, however, that the invention can be practiced without many of these details. Likewise, one skilled in the relevant technology will understand that the invention can include well-known structures or features that are not shown or described in detail, to avoid unnecessarily obscuring the descriptions of examples.

Wireless Communications System

[0021] FIG. **1** is a block diagram that illustrates a wireless telecommunication network **100** ("network **100**") in which aspects of the disclosed technology are incorporated. The network **100** includes base stations **102-1** through **102-4** (also referred to individually as "base station **102**" or collectively as "base stations **102**"). A base station is a type of network access node (NAN) that can also be referred to as a cell site, a base transceiver station, or a radio base station. The network **100** can include any combination of NANs including an access point, radio transceiver, gNodeB (gNB), NodeB, eNodeB (eNB), Home NodeB or Home eNodeB, or the like. In addition to being a wireless wide area network (WWAN) base station, a NAN can be a wireless local area network (WLAN) access point, such as an Institute of Electrical and Electronics Engineers (IEEE) 802.11 access point.

[0022] The NANs of a network **100** formed by the network **100** also include wireless devices **104-1** through **104-7** (referred to individually as "wireless device **104**" or collectively as "wireless devices **104**") and a core network **106**. The wireless devices **104** can correspond to or include network **100** entities capable of communication using various connectivity standards. For example, a 5G communication channel can use millimeter wave (mmW) access frequencies of 28 GHz or more. In some implementations, the wireless device **104** can operatively couple to a base station **102** over a long-term evolution/long-term evolution-advanced (LTE/LTE-A) communication channel, which is referred to as a 4G communication channel.

[0023] The core network **106** provides, manages, and controls security services, user authentication, access authorization, tracking, internet protocol (IP) connectivity, and other access, routing, or mobility functions. The base stations **102** interface with the core network **106** through a first set of backhaul links (e.g., S1 interfaces) and can perform radio configuration and scheduling for communication with the wireless devices **104** or can operate under the control of a base station controller (not shown). In some examples, the base stations **102** can communicate with each other, either directly or indirectly (e.g., through the core network **106**), over a second set of backhaul links **110-1** through **110-3** (e.g., X1 interfaces), which can be wired or wireless communication links.

[0024] The base stations **102** can wirelessly communicate with the wireless devices **104** via one or more base station antennas. The cell sites can provide communication coverage for geographic coverage areas **112-1** through **112-4** (also referred to individually as "coverage area **112**" or collectively as "coverage areas **112**"). The coverage area **112** for a base station **102** can be divided into sectors making up only a portion of the coverage area (not shown). The network **100** can include base stations of different types (e.g., macro and/or small cell base stations). In some implementations, there can be overlapping coverage areas **112** for different service environments (e.g., Internet of Things (IoT), mobile broadband (MBB), vehicle-to-everything (V2X), machine-to-machine (M2M), machine-to-everything (M2X), ultra-reliable low-latency communication (URLLC), machine-type communication (MTC), etc.).

[0025] The network **100** can include a 5G network **100** and/or an LTE/LTE-A or other network. In an LTE/LTE-A network, the term "eNBs" is used to describe the base stations **102**, and in 5G new radio (NR) networks, the term "gNBs" is used to describe the base stations **102** that can include mmW communications. The network **100** can thus form a heterogeneous network **100** in which different types of base stations provide coverage for various geographic regions. For example, each base station **102** can provide communication coverage for a macro cell, a small cell, and/or other types of cells. As used herein, the term "cell" can relate to a base station, a carrier or component carrier associated with the base station, or a coverage area (e.g., sector) of a carrier or base station, depending on context.

[0026] A macro cell generally covers a relatively large geographic area (e.g., several kilometers in radius) and can allow access by wireless devices that have service subscriptions with a wireless network **100** service provider. As indicated earlier, a small cell is a lower-powered base station, as compared to a macro cell, and can operate in the same or different (e.g., licensed, unlicensed) frequency bands as macro cells. Examples of small cells include pico cells, femto cells, and micro cells. In general, a pico cell can cover a relatively smaller geographic area and can allow unrestricted access by wireless devices that have service subscriptions with the network **100** provider. A femto cell covers a relatively smaller geographic area (e.g., a home) and can provide restricted access by wireless devices having an association with the femto unit (e.g., wireless devices in a closed subscriber group (CSG), wireless devices for users in the home). A base station can support one or multiple (e.g., two, three, four, and the like) cells (e.g., component carriers). All fixed transceivers noted herein that can provide access to the network **100** are NANs, including small cells.

[0027] The communication networks that accommodate various disclosed examples can be packet-based networks that operate according to a layered protocol stack. In the user plane, communications at the bearer or Packet Data Con-

vergence Protocol (PDCP) layer can be IP-based. A Radio Link Control (RLC) layer then performs packet segmentation and reassembly to communicate over logical channels. A Medium Access Control (MAC) layer can perform priority handling and multiplexing of logical channels into transport channels. The MAC layer can also use Hybrid ARQ (HARQ) to provide retransmission at the MAC layer, to improve link efficiency. In the control plane, the Radio Resource Control (RRC) protocol layer provides establishment, configuration, and maintenance of an RRC connection between a wireless device **104** and the base stations **102** or core network **106** supporting radio bearers for the user plane data. At the Physical (PHY) layer, the transport channels are mapped to physical channels.

[0028] Wireless devices can be integrated with or embedded in other devices. As illustrated, the wireless devices **104** are distributed throughout the network **100**, where each wireless device **104** can be stationary or mobile. For example, wireless devices can include handheld mobile devices **104-1** and **104-2** (e.g., smartphones, portable hotspots, tablets, etc.); laptops **104-3**; wearables **104-4**; drones **104-5**; vehicles with wireless connectivity **104-6**; head-mounted displays with wireless augmented reality/virtual reality (AR/VR) connectivity **104-7**; portable gaming consoles; wireless routers, gateways, modems, and other fixed-wireless access devices; wirelessly connected sensors that provide data to a remote server over a network; IoT devices such as wirelessly connected smart home appliances; etc.

[0029] A wireless device (e.g., wireless devices **104**) can be referred to as a user equipment (UE), a customer premises equipment (CPE), a mobile station, a subscriber station, a mobile unit, a subscriber unit, a wireless unit, a remote unit, a handheld mobile device, a remote device, a mobile subscriber station, a terminal equipment, an access terminal, a mobile terminal, a wireless terminal, a remote terminal, a handset, a mobile client, a client, or the like.

[0030] A wireless device can communicate with various types of base stations and network **100** equipment at the edge of a network **100** including macro eNBs/gNBs, small cell eNBs/gNBs, relay base stations, and the like. A wireless device can also communicate with other wireless devices either within or outside the same coverage area of a base station via device-to-device (D2D) communications.

[0031] The communication links **114-1** through **114-9** (also referred to individually as "communication link **114**" or collectively as "communication links **114**") shown in network **100** include uplink (UL) transmissions from a wireless device **104** to a base station **102** and/or downlink (DL) transmissions from a base station **102** to a wireless device **104**. The downlink transmissions can also be called forward link transmissions while the uplink transmissions can also be called reverse link transmissions. Each communication link **114** includes one or more carriers, where each carrier can be a signal composed of multiple sub-carriers (e.g., waveform signals of different frequencies) modulated according to the various radio technologies. Each modulated signal can be sent on a different sub-carrier and carry control information (e.g., reference signals, control channels), overhead information, user data, etc. The communication links **114** can transmit bidirectional communications using frequency division duplex (FDD) (e.g., using paired spectrum resources) or time division duplex (TDD) operation (e.g., using unpaired spectrum resources). In some implementations, the communication links **114** include LTE and/or mmW communication links.

[0032] In some implementations of the network **100**, the base stations **102** and/or the wireless devices **104** include multiple antennas for employing antenna diversity schemes to improve communication quality and reliability between base stations **102** and wireless devices **104**. Additionally or alternatively, the base stations **102** and/or the wireless devices **104** can employ multiple-input, multiple-output (MIMO) techniques that can take advantage of multi-path environments to transmit multiple spatial layers carrying the same or different coded data.

[0033] In some examples, the network **100** implements 6G technologies including increased densification or diversification of network nodes. The network **100** can enable terrestrial and non-terrestrial transmissions. In this context, a Non-Terrestrial Network (NTN) is enabled by one or more satellites, such as satellites **116-1** and **116-2**, to deliver services anywhere and anytime and provide coverage in areas that are unreachable by any conventional Terrestrial Network (TN). A 6G implementation of the network **100** can support terahertz (THz) communications. This can support wireless applications that demand ultrahigh quality of service (QoS) requirements and multi-terabits-per-second data transmission in the era of 6G and beyond, such as terabit-per-second backhaul systems, ultra-high-definition content streaming among mobile devices, AR/VR, and wireless high-bandwidth secure communications. In another example of 6G, the network **100** can implement a converged Radio Access Network (RAN) and Core architecture to achieve Control and User Plane Separation (CUPS) and achieve extremely low user plane latency. In yet another example of 6G, the network **100** can implement a converged Wi-Fi and Core architecture to increase and improve indoor coverage.

Transformer for Neural Network

[0034] To assist in understanding the present disclosure, some concepts relevant to neural networks and machine learning (ML) are discussed herein. Generally, a neural network comprises a number of computation units (sometimes referred to as "neurons"). Each neuron receives an input value and applies a function to the input to generate an output value. The function typically includes a parameter (also referred to as a "weight") whose value is learned through the process of training. A plurality of neurons may be organized into a neural network layer (or simply "layer") and there may be multiple such layers in a neural network. The output of one layer may be provided as input to a subsequent layer. Thus, input to a neural network may be processed through a succession of layers until an output of the neural network is generated by a final layer. This is a simplistic discussion of neural networks and there may be more complex neural network designs that include feedback connections, skip connections, and/or other such possible connections between neurons and/or layers, which are not discussed in detail here.

[0035] A deep neural network (DNN) is a type of neural network having multiple layers and/or a large number of neurons. The term DNN may encompass any neural network having multiple layers, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), multilayer perceptrons (MLPs), Generative Adversarial Networks

(GANs), Variational Autoencoders (VAEs), and Auto-regressive Models, among others.

[0036] DNNs are often used as ML-based models for modeling complex behaviors (e.g., human language, image recognition, object classification) in order to improve the accuracy of outputs (e.g., more accurate predictions) such as, for example, as compared with models with fewer layers. In the present disclosure, the term "ML-based model" or more simply "ML model" may be understood to refer to a DNN. Training an ML model refers to a process of learning the values of the parameters (or weights) of the neurons in the layers such that the ML model is able to model the target behavior to a desired degree of accuracy. Training typically requires the use of a training dataset, which is a set of data that is relevant to the target behavior of the ML model.

[0037] As an example, to train an ML model that is intended to model human language (also referred to as a language model), the training dataset may be a collection of text documents, referred to as a text corpus (or simply referred to as a corpus). The corpus may represent a language domain (e.g., a single language), a subject domain (e.g., scientific papers), and/or may encompass another domain or domains, be they larger or smaller than a single language or subject domain. For example, a relatively large, multilingual and non-subject-specific corpus may be created by extracting text from online webpages and/or publicly available social media posts. Training data may be annotated with ground truth labels (e.g., each data entry in the training dataset may be paired with a label), or may be unlabeled.

[0038] Training an ML model generally involves inputting into an ML model (e.g., an untrained ML model) training data to be processed by the ML model, processing the training data using the ML model, collecting the output generated by the ML model (e.g., based on the inputted training data), and comparing the output to a desired set of target values. If the training data is labeled, the desired target values may be, e.g., the ground truth labels of the training data. If the training data is unlabeled, the desired target value may be a reconstructed (or otherwise processed) version of the corresponding ML model input (e.g., in the case of an autoencoder), or can be a measure of some target observable effect on the environment (e.g., in the case of a reinforcement learning agent). The parameters of the ML model are updated based on a difference between the generated output value and the desired target value. For example, if the value outputted by the ML model is excessively high, the parameters may be adjusted so as to lower the output value in future training iterations. An objective function is a way to quantitatively represent how close the output value is to the target value. An objective function represents a quantity (or one or more quantities) to be optimized (e.g., minimize a loss or maximize a reward) in order to bring the output value as close to the target value as possible. The goal of training the ML model typically is to minimize a loss function or maximize a reward function.

[0039] The training data may be a subset of a larger data set. For example, a data set may be split into three mutually exclusive subsets: a training set, a validation (or cross-validation) set, and a testing set. The three subsets of data may be used sequentially during ML model training. For example, the training set may be first used to train one or more ML models, each ML model, e.g., having a particular architecture, having a particular training procedure, being describable by a set of model hyperparameters, and/or otherwise being varied from the other of the one or more ML models. The validation (or cross-validation) set may then be used as input data into the trained ML models to, e.g., measure the performance of the trained ML models and/or compare performance between them. Where hyperparameters are used, a new set of hyperparameters may be determined based on the measured performance of one or more of the trained ML models, and the first step of training (i.e., with the training set) may begin again on a different ML model described by the new set of determined hyperparameters. In this way, these steps may be repeated to produce a more performant trained ML model. Once such a trained ML model is obtained (e.g., after the hyperparameters have been adjusted to achieve a desired level of performance), a third step of collecting the output generated by the trained ML model applied to the third subset (the testing set) may begin. The output generated from the testing set may be compared with the corresponding desired target values to give a final assessment of the trained ML model's accuracy. Other segmentations of the larger data set and/or schemes for using the segments for training one or more ML models are possible.

[0040] Backpropagation is an algorithm for training an ML model. Backpropagation is used to adjust (also referred to as update) the value of the parameters in the ML model, with the goal of optimizing the objective function. For example, a defined loss function is calculated by forward propagation of an input to obtain an output of the ML model and a comparison of the output value with the target value. Backpropagation calculates a gradient of the loss function with respect to the parameters of the ML model, and a gradient algorithm (e.g., gradient descent) is used to update (i.e., "learn") the parameters to reduce the loss function. Backpropagation is performed iteratively so that the loss function is converged or minimized. Other techniques for learning the parameters of the ML model may be used. The process of updating (or learning) the parameters over many iterations is referred to as training. Training may be carried out iteratively until a convergence condition is met (e.g., a predefined maximum number of iterations has been performed, or the value outputted by the ML model is sufficiently converged with the desired target value), after which the ML model is considered to be sufficiently trained. The values of the learned parameters may then be fixed and the ML model may be deployed to generate output in real-world applications (also referred to as "inference").

[0041] In some examples, a trained ML model may be fine-tuned, meaning that the values of the learned parameters may be adjusted slightly in order for the ML model to better model a specific task. Fine-tuning of an ML model typically involves further training the ML model on a number of data samples (which may be smaller in number/cardinality than those used to train the model initially) that closely target the specific task. For example, an ML model for generating natural language that has been trained generically on publically-available text corpora may be, e.g., fine-tuned by further training using specific training samples. The specific training samples can be used to generate language in a certain style or in a certain format. For example, the ML model can be trained to generate a blog post having a particular style and structure with a given topic.

[0042] Some concepts in ML-based language models are now discussed. It may be noted that, while the term "language model" has been commonly used to refer to a ML-

based language model, there could exist non-ML language models. In the present disclosure, the term "language model" may be used as shorthand for an ML-based language model (i.e., a language model that is implemented using a neural network or other ML architecture), unless stated otherwise. For example, unless stated otherwise, the "language model" encompasses LLMs.

[0043] A language model may use a neural network (typically a DNN) to perform natural language processing (NLP) tasks. A language model may be trained to model how words relate to each other in a textual sequence, based on probabilities. A language model may contain hundreds of thousands of learned parameters or in the case of a large language model (LLM) may contain millions or billions of learned parameters or more. As non-limiting examples, a language model can generate text, translate text, summarize text, answer questions, write code (e.g., Phyton, JavaScript, or other programming languages), classify text (e.g., to identify spam emails), create content for various purposes (e.g., social media content, factual content, or marketing content), or create personalized content for a particular individual or group of individuals. Language models can also be used for chatbots (e.g., virtual assistance).

[0044] In recent years, there has been interest in a type of neural network architecture, referred to as a transformer, for use as language models. For example, the Bidirectional Encoder Representations from Transformers (BERT) model, the Transformer-XL model, and the Generative Pre-trained Transformer (GPT) models are types of transformers. A transformer is a type of neural network architecture that uses self-attention mechanisms in order to generate predicted output based on input data that has some sequential meaning (i.e., the order of the input data is meaningful, which is the case for most text input). Although transformer-based language models are described herein, it should be understood that the present disclosure may be applicable to any ML-based language model, including language models based on other neural network architectures such as recurrent neural network (RNN)-based language models.

[0045] FIG. **2** is a block diagram of an example transformer **212**. A transformer is a type of neural network architecture that uses self-attention mechanisms to generate predicted output based on input data that has some sequential meaning (i.e., the order of the input data is meaningful, which is the case for most text input). Self-attention is a mechanism that relates different positions of a single sequence to compute a representation of the same sequence. Although transformer-based language models are described herein, it should be understood that the present disclosure may be applicable to any machine learning (ML)-based language model, including language models based on other neural network architectures such as recurrent neural network (RNN)-based language models.

[0046] The transformer **212** includes an encoder **208** (which can comprise one or more encoder layers/blocks connected in series) and a decoder **210** (which can comprise one or more decoder layers/blocks connected in series). Generally, the encoder **208** and the decoder **210** each include a plurality of neural network layers, at least one of which can be a self-attention layer. The parameters of the neural network layers can be referred to as the parameters of the language model.

[0047] The transformer **212** can be trained to perform certain functions on a natural language input. For example, the functions include summarizing existing content, brainstorming ideas, writing a rough draft, fixing spelling and grammar, and translating content. Summarizing can include extracting key points from an existing content in a high-level summary. Brainstorming ideas can include generating a list of ideas based on provided input. For example, the ML model can generate a list of names for a startup or costumes for an upcoming party. Writing a rough draft can include generating writing in a particular style that could be useful as a starting point for the user's writing. The style can be identified as, e.g., an email, a blog post, a social media post, or a poem. Fixing spelling and grammar can include correcting errors in an existing input text. Translating can include converting an existing input text into a variety of different languages. In some embodiments, the transformer **212** is trained to perform certain functions on other input formats than natural language input. For example, the input can include objects, images, audio content, or video content, or a combination thereof.

[0048] The transformer **212** can be trained on a text corpus that is labeled (e.g., annotated to indicate verbs, nouns) or unlabeled. Large language models (LLMs) can be trained on a large unlabeled corpus. The term "language model," as used herein, can include an ML-based language model (e.g., a language model that is implemented using a neural network or other ML architecture), unless stated otherwise. Some LLMs can be trained on a large multi-language, multi-domain corpus to enable the model to be versatile at a variety of language-based tasks such as generative tasks (e.g., generating human-like natural language responses to natural language input). FIG. **2** illustrates an example of how the transformer **212** can process textual input data. Input to a language model (whether transformer-based or otherwise) typically is in the form of natural language that can be parsed into tokens. It should be appreciated that the term "token" in the context of language models and Natural Language Processing (NLP) has a different meaning from the use of the same term in other contexts such as data security. Tokenization, in the context of language models and NLP, refers to the process of parsing textual input (e.g., a character, a word, a phrase, a sentence, a paragraph) into a sequence of shorter segments that are converted to numerical representations referred to as tokens (or "compute tokens"). Typically, a token can be an integer that corresponds to the index of a text segment (e.g., a word) in a vocabulary dataset. Often, the vocabulary dataset is arranged by frequency of use. Commonly occurring text, such as punctuation, can have a lower vocabulary index in the dataset and thus be represented by a token having a smaller integer value than less commonly occurring text. Tokens frequently correspond to words, with or without white space appended. In some examples, a token can correspond to a portion of a word.

[0049] For example, the word "greater" can be represented by a token for [great] and a second token for [er]. In another example, the text sequence "write a summary" can be parsed into the segments [write], [a], and [summary], each of which can be represented by a respective numerical token. In addition to tokens that are parsed from the textual sequence (e.g., tokens that correspond to words and punctuation), there can also be special tokens to encode non-textual information. For example, a [CLASS] token can be a special token that corresponds to a classification of the textual sequence (e.g., can classify the textual sequence as a list, a

paragraph), an [EOT] token can be another special token that indicates the end of the textual sequence, other tokens can provide formatting information, etc.

[0050] In FIG. **2**, a short sequence of tokens **202** corresponding to the input text is illustrated as input to the transformer **212**. Tokenization of the text sequence into the tokens **202** can be performed by some pre-processing tokenization module such as, for example, a byte-pair encoding tokenizer (the "pre" referring to the tokenization occurring prior to the processing of the tokenized input by the LLM), which is not shown in FIG. **2** for simplicity. In general, the token sequence that is inputted to the transformer **212** can be of any length up to a maximum length defined based on the dimensions of the transformer **212**. Each token **202** in the token sequence is converted into an embedding vector **206** (also referred to simply as an embedding **206**). An embedding **206** is a learned numerical representation (such as, for example, a vector) of a token that captures some semantic meaning of the text segment represented by the token **202**. The embedding **206** represents the text segment corresponding to the token **202** in a way such that embeddings corresponding to semantically related text are closer to each other in a vector space than embeddings corresponding to semantically unrelated text. For example, assuming that the words "write," "a," and "summary" each correspond to, respectively, a "write" token, an "a" token, and a "summary" token when tokenized, the embedding **206** corresponding to the "write" token will be closer to another embedding corresponding to the "jot down" token in the vector space as compared to the distance between the embedding **206** corresponding to the "write" token and another embedding corresponding to the "summary" token.

[0051] The vector space can be defined by the dimensions and values of the embedding vectors. Various techniques can be used to convert a token **202** to an embedding **206**. For example, another trained ML model can be used to convert the token **202** into an embedding **206**. In particular, another trained ML model can be used to convert the token **202** into an embedding **206** in a way that encodes additional information into the embedding **206** (e.g., a trained ML model can encode positional information about the position of the token **202** in the text sequence into the embedding **206**). In some examples, the numerical value of the token **202** can be used to look up the corresponding embedding in an embedding matrix **204** (which can be learned during training of the transformer **212**).

[0052] The generated embeddings **206** are input into the encoder **208**. The encoder **208** serves to encode the embeddings **206** into feature vectors **214** that represent the latent features of the embeddings **206**. The encoder **208** can encode positional information (i.e., information about the sequence of the input) in the feature vectors **214**. The feature vectors **214** can have very high dimensionality (e.g., on the order of thousands or tens of thousands), with each element in a feature vector **214** corresponding to a respective feature. The numerical weight of each element in a feature vector **214** represents the importance of the corresponding feature. The space of all possible feature vectors **214** that can be generated by the encoder **208** can be referred to as the latent space or feature space.

[0053] Conceptually, the decoder **210** is designed to map the features represented by the feature vectors **214** into meaningful output, which can depend on the task that was assigned to the transformer **212**. For example, if the transformer **212** is used for a translation task, the decoder **210** can map the feature vectors **214** into text output in a target language different from the language of the original tokens **202**. Generally, in a generative language model, the decoder **210** serves to decode the feature vectors **214** into a sequence of tokens. The decoder **210** can generate output tokens **216** one by one. Each output token **216** can be fed back as input to the decoder **210** in order to generate the next output token **216**. By feeding back the generated output and applying self-attention, the decoder **210** is able to generate a sequence of output tokens **216** that has sequential meaning (e.g., the resulting output text sequence is understandable as a sentence and obeys grammatical rules). The decoder **210** can generate output tokens **216** until a special [EOT] token (indicating the end of the text) is generated. The resulting sequence of output tokens **216** can then be converted to a text sequence in post-processing. For example, each output token **216** can be an integer number that corresponds to a vocabulary index. By looking up the text segment using the vocabulary index, the text segment corresponding to each output token **216** can be retrieved, the text segments can be concatenated together, and the final output text sequence can be obtained.

[0054] In some examples, the input provided to the transformer **212** includes instructions to perform a function on an existing text. In some examples, the input provided to the transformer includes instructions to perform a function on an existing text. The output can include, for example, a modified version of the input text and instructions to modify the text. The modification can include summarizing, translating, correcting grammar or spelling, changing the style of the input text, lengthening or shortening the text, or changing the format of the text. For example, the input can include the question "What is the weather like in Australia?" and the output can include a description of the weather in Australia.

[0055] Although a general transformer architecture for a language model and its theory of operation have been described above, this is not intended to be limiting. Existing language models include language models that are based only on the encoder of the transformer or only on the decoder of the transformer. An encoder-only language model encodes the input text sequence into feature vectors that can then be further processed by a task-specific layer (e.g., a classification layer). BERT is an example of a language model that can be considered to be an encoder-only language model. A decoder-only language model accepts embeddings as input and can use auto-regression to generate an output text sequence. Transformer-XL and GPT-type models can be language models that are considered to be decoder-only language models.

[0056] Because GPT-type language models tend to have a large number of parameters, these language models can be considered LLMs. An example of a GPT-type LLM is GPT-3. GPT-3 is a type of GPT language model that has been trained (in an unsupervised manner) on a large corpus derived from documents available to the public online. GPT-3 has a very large number of learned parameters (on the order of hundreds of billions), is able to accept a large number of tokens as input (e.g., up to 2,048 input tokens), and is able to generate a large number of tokens as output (e.g., up to 2,048 tokens). GPT-3 has been trained as a generative model, meaning that it can process input text sequences to predictively generate a meaningful output text sequence. ChatGPT is built on top of a GPT-type LLM and

has been fine-tuned with training datasets based on text-based chats (e.g., chatbot conversations). ChatGPT is designed for processing natural language, receiving chat-like inputs, and generating chat-like outputs.

[0057] A computer system can access a remote language model (e.g., a cloud-based language model), such as ChatGPT or GPT-3, via a software interface (e.g., an API). Additionally or alternatively, such a remote language model can be accessed via a network such as, for example, the Internet. In some implementations, such as, for example, potentially in the case of a cloud-based language model, a remote language model can be hosted by a computer system that can include a plurality of cooperating (e.g., cooperating via a network) computer systems that can be in, for example, a distributed arrangement. Notably, a remote language model can employ a plurality of processors (e.g., hardware processors such as, for example, processors of cooperating computer systems). Indeed, processing of inputs by an LLM can be computationally expensive/can involve a large number of operations (e.g., many instructions can be executed/large data structures can be accessed from memory), and providing output in a required timeframe (e.g., real time or near real time) can require the use of a plurality of processors/cooperating computing devices as discussed above.

[0058] Inputs to an LLM can be referred to as a prompt, which is a natural language input that includes instructions to the LLM to generate a desired output. A computer system can generate a prompt that is provided as input to the LLM via its API. As described above, the prompt can optionally be processed or pre-processed into a token sequence prior to being provided as input to the LLM via its API. A prompt can include one or more examples of the desired output, which provides the LLM with additional information to enable the LLM to generate output according to the desired output. Additionally or alternatively, the examples included in a prompt can provide inputs (e.g., example inputs) corresponding to/as can be expected to result in the desired outputs provided. A one-shot prompt refers to a prompt that includes one example, and a few-shot prompt refers to a prompt that includes multiple examples. A prompt that includes no examples can be referred to as a zero-shot prompt.

Cloud Solution Deployment System

[0059] FIG. **3** illustrates an embodiment of the system **300** for selecting and implementing a new cloud solution. Data **302** can include a specification for a cloud solution. For example, the type of cloud solution can include infrastructure-as-a-service, platform-as-a-service, software-as-a-service, or function-as-a-service. The system **300** can determine a list of cloud service providers **304**. The list of cloud service providers can include the cloud infrastructure **306** that the cloud service provider can use to host the cloud solution. In one embodiment, the system **300** can prioritize high-efficiency cloud infrastructure. For example, high-efficiency cloud infrastructure can include high-efficiency components. High-efficiency components can generate the same processing power while simultaneously consuming less energy. Therefore, the amount of greenhouse gases generated by the system can be reduced because of the high-efficiency components that use less energy.

[0060] The system **300** can analyze the list of cloud service providers **304** and other requirements for the cloud service contained in the data **302** and input the analysis into a cloud ranking system **308**. The cloud ranking system **308** can rank the cloud service providers based on an ability to host the cloud solution and meet any other given requirements for the cloud solution. A cloud service provider ranking list **310** can be generated by the cloud ranking system **308**. For example, the cloud service providers can be ranked based on a series of criteria such as cost to host the cloud solution, security features offered by the cloud service provider, cloud infrastructure **306** components, an ability of the cloud infrastructure **306** to support a threshold user traffic volume or user traffic type, the provisioning speed of the cloud infrastructure **306**, reliability of the cloud infrastructure **306**, or scalability of the cloud solution on the cloud infrastructure **306**. In one example, calculating a metric score can determine the cloud service provider ranking list **310**. The metric score can be calculated by assigning a weighted value to each criterion and averaging the values. In another example, the cloud service provider ranking list **310** can be determined by assigning a weighted value to a selected criterion and ranking the cloud service providers with the selected criterion. The cloud service provider with the highest value for the selected criterion can be ranked the highest.

[0061] A cloud service provider can be selected from the cloud service provider ranking list **310**. For example, a cloud service provider with a high ranking can be selected. In another example, a cloud service provider can be selected based on specific criteria such as cost, a specific cloud infrastructure **306** component, or an ability to meet a given requirement specified in the cloud solution specification. When a cloud service provider is selected, an implementation procedure can be generated based on the specification of the cloud service provider's Application Programming Interface (API) contained in the data **302**. An LLM can be used to deploy the cloud solution on the cloud infrastructure **306** of the selected cloud service provider. When the cloud solution has been deployed, a user computer **312** can access the cloud solution over a network such as a 5G or Wi-Fi network.

[0062] FIG. **4** illustrates a flow diagram **400** for selecting a new cloud provider for an existing cloud solution. At step **402**, the management console allows a user to manage the existing cloud solution. For example, the user can select an appropriate cloud workload or the number of instances needed to manage the expected transactions of the cloud solution effectively. At step **404**, a common cloud language can be generated based on the user's selections and the specification for the cloud solution. For example, the common cloud language can be generated in a generic programming language that can be converted to a programming language and syntax used by a cloud service provider. An LLM can generate the common cloud language and can convert the common cloud language to the programming language and syntax used by the selected cloud service provider.

[0063] Generating the cloud solution in a common cloud language allows for the cloud solution to be more easily deployed on new cloud infrastructure while preventing a multitude of common challenges associated with cloud technology. For example, the common cloud language can prevent vendor lock-in, which is a phenomenon where a cloud solution customer does not migrate the cloud solution to a different cloud service provider due to the perceived complexity and cost of the migration, by reducing the cost

and complexity of the migration. A user can migrate the cloud solution when cost savings opportunities arise. The common cloud language can help prevent cost optimization shortcomings by providing updated pricing information for different cloud service providers while allowing the cloud solution to be easily migrated to a lower-cost cloud service provider.

[0064] At step **406**, the cloud service provider discovery process can occur. The discovery process can include searching for cloud service providers that meet the requirements of the cloud solution. At step **408**, a cost calculation for each discovered cloud service provider can occur. For example, the cost calculation can include the cost to host and maintain the cloud solution on the cloud service provider's cloud infrastructure. The cost calculation can be a monthly cost or a yearly cost. In another example, the cost calculation can be normalized based on the cloud service provider's cloud infrastructure. Normalizing the cost calculation allows for a more accurate comparison between the different service providers when the cloud service providers do not have identical cloud infrastructure.

[0065] At step **410**, the requirements of the cloud service provider to deploy the cloud solution can be determined for each cloud service provider. At step **412**, the cloud service providers can be compared. The cloud service providers can be compared based on multiple factors such as cost, security features offered by the cloud service provider, cloud infrastructure components, the ability of the cloud infrastructure to support a threshold user traffic volume or user traffic type, the provisioning speed of the cloud infrastructure, reliability of the cloud infrastructure, or scalability of the cloud solution on the cloud infrastructure. Additionally, the cloud service providers can be compared based on the requirements each cloud service provider has to implement a cloud solution on their cloud infrastructure.

[0066] At step **414**, the cloud service providers can be ranked based on the comparison. For example, the cloud service provider with the highest overall ranking for each factor used in the comparison can be ranked the highest. In another example, the highest-ranking cloud service provider can be the cloud service provider with the highest ranking for one factor, such as cost or security features. A cloud service provider can be selected based on the ranking where, for example, the highest-ranking or a high-ranking cloud service provider is selected. At step **416**, the cloud solution can be migrated from a current cloud service provider to the selected cloud service provider.

[0067] At step **418**, a continuous discovery process is performed. For example, the continuous discovery process can include determining updates to the cloud service provider's pricing structure or cloud infrastructure. At step **420**, the cost of the cloud solution and the requirement of the cloud solution can be monitored. For example, the cloud solution can be monitored for an increase in the price to host the cloud solution of the current cloud service provider. The cloud service provider's ability to meet a given requirement of the cloud solution can also be monitored. When a current cloud service provider fails to meet a given requirement or when a price increase beyond a certain threshold occurs, the process can begin again at step **406**.

[0068] FIG. **5** illustrates a block diagram **500** of the various features of an embodiment of the system. The cloud service provider box **564** can contain the various cloud service providers hosting the different cloud solutions controlled by a user. For example, Amazon Web Services (AWS) can host account X. The system can use an AWS calculator API **502** to determine AWS's unique programming syntax used to deploy and maintain the cloud solution for account X. Google Cloud Platform (GCP) can host account Y. The system can use a GCP calculator API **504** to determine GCP's unique programming syntax used to deploy and maintain the cloud solution for account Y. Azure can host account Z. The system can use an Azure calculator API **506** to determine Azure's unique programming syntax used to deploy and maintain the cloud solution for account Z.

[0069] The discovery box **566** can be the processes and interactions for the user to implement or redeploy a cloud solution. The user can leverage a console to articulate their requirements, such as a cloud service provider preference, a new workload to be deployed, or a cost requirement. The user can use the hardware discovery process **508** to determine the hardware requirements of the cloud solution or to determine the hardware components offered in the cloud infrastructure of a cloud service provider. The new deployment monitoring **510** can monitor the cloud service providers to determine requirements for deployment or migration of the cloud solution. The real-time cost analysis **512** can monitor the cloud service providers to determine the deployment or migration cost of the cloud solution and/or the hosting cost of the cloud solution. This can allow a user to know the process and cost of deploying the cloud solution or the cost of hosting the cloud solution for multiple cloud service providers. Additionally, this aids in sizing the cloud infrastructure, workloads, and pricing to align with the user's needs or requirements.

[0070] The component box **568** can be the cloud infrastructure components used to host and execute the cloud solution. For example, a cloud service provider's cloud infrastructure components can include a virtual machine **514**, storage **516**, a load balancer **518**, a database **520**, a container **522**, and networking components **524**. The cloud infrastructure components can be unique to each cloud service provider. For example, AWS can have different cloud infrastructure components than GCP or Azure. Additionally, the different cloud service providers can use a unique naming nomenclature or terminology for the different cloud infrastructure components, causing the same cloud infrastructure components to have different names for the each cloud service provider. Additionally, the different cloud infrastructure components can be supported by different APIs. However, the system can control for the differences in terminology used by the cloud service providers by converting the terminology to a unified component terminology that allows a user to use consistent terminology when interacting with the different cloud service providers.

[0071] The execution platform box **570** can contain and control the core operations and tools of the system. For example, the execution platform box **570** can contain the calculator **526**. The calculator **526** can integrate with cost calculation APIs from multiple cloud service providers to enable precise cost assessments for various configurations and workloads implemented on different cloud infrastructure. For example, calculator **526** can calculate the cost of hosting a cloud solution with a virtual machine (VM) with a specific configuration, such as 2 CPUs and 4 gigabytes of memory, for a day or week on the cloud service provider's cloud infrastructure. Multiplying the cost per VM by the

total number of VMs can show the cloud solution's daily, weekly, monthly, or yearly operational costs.

[0072] The cost comparison tool **528** can compare the hosting cost of the cloud solution for the different cloud service providers. For example, the cost comparison tool **528** can account for differences in the cloud infrastructure components when comparing costs. The cost monitoring tool **530** can monitor the costs of different cloud service providers to determine if migrating the cloud solution to a different cloud service provider would generate a cost savings for the user. The integration tool **532** can determine the processes and requirements to integrate the cloud solution on the cloud infrastructure of a cloud service provider. The code completion tool **534** can determine the code required to deploy the cloud solution. Additionally, the code completion tool **534** can monitor the code generated by the code generation tool **544** to determine whether the code complies with the required criteria of the cloud service provider. The code generation tool **544** can generate the code required to deploy the cloud solution. In one example, code completion tool **534** and the code generation tool **544** can use an LLM to generate the required code.

[0073] The cross-cloud migration tool **536** can monitor and control the migration of the cloud solution from one cloud service provider to a different cloud service provider. The inner cloud migration tool **538** can monitor and control the migration of the cloud solution to different cloud infrastructure of the same cloud service provider currently hosting the cloud solution. The security tool **540** can determine the security protocols and features offered by the different cloud service providers. The security tool **540** can monitor the cloud solution for security risks and vulnerabilities and can notify the user if a cyberattack or security threat is detected. The traffic monitoring tool **542** can monitor the amount of user traffic volume and user traffic type of the cloud solution. Combining the data generated from the different tools, such as user traffic volume and type, with user-defined configurations assists the system in recommending the most suitable cloud infrastructure and migration strategies.

[0074] The intelligent platform box **572** can provide plain-language recommendations to the user based on the data and recommendations received from the execution platform. The intelligent platform box **572** can use an LLM **546** to generate recommendations. The cost recommendation tool **548** can use the LLM **546** to generate a recommended cost to host the cloud solution. For example, the cost recommendation tool **548** can display to the user the expected cost to host the cloud solution for a given time period or the average cost to host the cloud solution. For example, the average cost to host the cloud solution can be calculated by averaging the cost calculation of each cloud service provider.

[0075] The code recommendation tool **550** can display to the user the recommended code used to deploy the cloud solution on the cloud service provider's cloud infrastructure. The hardware usage intent recognition tool **552** can recommend the different cloud infrastructure components to a user based on the stated intent of the user and the execution platform tools.

[0076] The cross-cloud configuration box **574** can act as the central portal for users to input the requirements for the cloud solution using the unified component terminology. The budget configuration tool **554** can allow a user to input a given budget. For example, a user can specify a minimum or maximum budget for the cloud solution. The cloud vendor preference tool **556** allows a user to specify a preferred cloud vendor. For example, a user may prefer to use either AWS or GCP and not Azure, and the cloud vendor preference tool **556** allows the user to input this preference. The automation approval workflow tool **558** can allow the user to approve the actions taken by the system. Approving the system's actions can allow the user to supervise the system's actions and prevent any unwanted actions. The reporting and dashboard **560** can display information relating to the cloud solution, such as the selected cloud service provider, cloud infrastructure components, or cost. The telemetry and monitoring tool **562** can display the status of the cloud solution. For example, the telemetry and monitoring tool **562** can indicate that the cloud solution is functioning correctly or that an error must be addressed. The cross-cloud configuration can allow for the management of multiple cloud solutions and allows the user to make real-time data-driven decisions about the cloud solution.

[0077] FIG. **6** illustrates a flow diagram of an embodiment of a calculator tool **600** for the system. The calculator tool **600** can conduct cost calculations for the different cloud infrastructure offered by different cloud service providers. At step **602**, the cloud configuration checker can receive the determined or specified cloud configuration needed to run and host the cloud solution. For example, the cloud configuration checker can receive an indication of the chosen or needed cloud infrastructure components. At step **604**, a filter configuration can be applied. For example, at step **606**, the filter configuration can receive multiple filter criteria. The filter criteria can include a region, instance type, tenancy, volume type, and/or memory type or amount.

[0078] At step **608**, the cloud service provider API integration can occur. The cloud service provider API integration allows the API of the system to communicate and receive data and information from the API from a cloud service provider. Converting the terminology can allow the system to determine the exact cloud infrastructure components that will be used for a given cloud service provider, which can allow the system to calculate a more accurate cost calculation. At step **610**, the output can be parsed to convert it to a more suitable format for the calculator tool **600** to use during the cost calculation.

[0079] At step **612**, the hardware matching tool can match the unified terminology used to select the cloud infrastructure with the unique language used by a cloud service provider. At step **614**, the cost calculation can occur. The cost calculation determines the final cost for the cloud solution to be run and hosted by a cloud service provider. The cost calculation can be for the cloud solution's daily, weekly, monthly, or yearly cost. At step **616**, a cost comparison matrix can be outputted. The cost comparison matrix can include the cost calculation for multiple cloud service providers. In one example, the cost comparison matrix is displayed to the user.

[0080] FIG. **7** illustrates a flow diagram for an embodiment of the recommendation process **700** for recommending a cloud service provider. For example, the recommendation process **700** can recommend changing cloud service providers or not changing cloud service providers. The recommendation process can also recommend changing the cloud infrastructure used to host the cloud solution to improve pricing and/or efficiency. The recommendation can be based on a variety of different types of data. For example, the

recommendation process **700** can factor in existing cost input **708**, new cost input **710**, the number of instances **712**, and/or telemetry and monitoring data **702**. The telemetry and monitoring data **702** can include current traffic metrics **714** and/or current hardware utilization metrics **716**. The recommendation process **700** can also factor in component metrics **704**, such as equivalent replacement components **718**, supported capabilities **720**, and/or networking and security **722**.

[0081] An evaluation engine **724** can evaluate the different types of data using an LLM **726** and can input the evaluation into the hardware recommendation **728**. The LLM **726** can use algorithms to identify suitable replacement components. The hardware recommendation **728** can recommend different cloud infrastructure components to use for the cloud solution. The hardware recommendation **728** can recommend no change **732** to the cloud infrastructure. The hardware recommendation **728** can recommend replacement **730** of the cloud infrastructure. When the hardware recommendation **728** recommends a replacement **730**, the replacement can cause a recommendation for new cloud infrastructure on the same cloud **734** or new cloud infrastructure on a different cloud **736**. For example, a different cloud **736** can be a cloud solution offered by a cloud service provider that is different from the cloud service provider currently hosting the cloud solution.

[0082] By performing the recommendation process **700**, the user can know which cloud infrastructure components are currently active, the specifications of the components to facilitate alignment with newer and/or different alternatives, and assess how effectively the cloud infrastructure is being utilized. The recommendation process **700** can also give insights into the functionality of current components.

[0083] FIG. **8** illustrates a flow diagram of an embodiment **800** of the monitoring hub **802**. The monitoring hub **802** can monitor price changes **804** and/or cloud API changes **816**. For example, user interaction or approval can only be required when a price or API change occurs.

[0084] When a price change **804** occurs, there can be an impact on the components **806**. For example, the cost to maintain the cloud solution on the components can decrease, allowing for cost savings for the user. The decrease in cost can also occur for a different cloud service provider, causing a cost impact on the components **806**. At step **808**, a recalculation of the component cost for the current cloud can occur. At step **810**, the component cost for the new cloud is recalculated. At step **812**, the monitoring hub can generate a comparison matrix output, which can include the recalculations for the current and new cloud. At step **814**, the monitoring can generate a cost comparison based on the comparison matrix. A user can use the cost comparison to approve a change to the cloud solution, such as approving a change to move the cloud solution to a different cloud service provider.

[0085] When a cloud API change **816** occurs, existing code can need updating to keep the cloud solution functioning correctly. At step **818**, the monitoring hub **802** can check for impacted component code. At step **820**, the monitoring hub **802** can use the LLM to check for new component code. At step **822**, the monitoring hub **802** can receive replacement code that can be used for a cost calculation. At step **824**, the monitoring hub **802** can run a cost calculation on the existing code. At step **826**, the monitoring hub **802** can run a cost calculation on the replacement code. At step **828**, the monitoring hub **802** can output a comparison of the cost calculation on the existing code and the replacement code. At step **830**, the monitoring hub **802** can determine if the output matches. For example, the outputs can match when the cost comparison is the same, and the cloud solution functions similarly with the replacement code and existing code. At step **832**, the output matches, and the monitoring hub **802** can deploy the new replacement code. At step **834**, the output does not match, and the monitoring hub **802** can notify the user or a support system to check the outputs manually. Generating new code for changes in the API specification allows for continuous integration and continuous deployment processes to be followed, which can ensure the cloud solution is not affected by evolving API requirements.

[0086] FIG. **9** illustrates a flow diagram of an embodiment **900** for replacing a cloud infrastructure component without impacting end-user traffic. At step **902**, a determination can be made to select a new component. Depending on the type of component selected, a different step can occur. For example, step **904** can occur when a source component is selected, and step **906** can occur when a target component is selected. At step **908**, a determination can be made as to whether the selected new component will be implemented on the same cloud. When the implementation does not occur on the same cloud, step **912** can occur. At step **912**, the domain name changes process can occur. When the domain name changes process has occurred, step **910** can occur. Additionally, step **910** can occur when the selected new component is determined to be implemented on the same cloud. At step **910**, the new component is deployed.

[0087] At step **914**, the traffic can be migrated instance by instance onto the newly deployed component. For example, migrating the traffic instance by instance can help prevent large-scale issues by only allowing minimal traffic on the new component to determine if it was correctly deployed with no errors. Migrating traffic this way becomes more important when the new component is deployed on a new cloud as more components need to be migrated and errors can be more frequent. At step **916**, the traffic can be monitored along with the error state. At step **918**, the system can determine if the deployment of the new component was a success. When an error occurs, the system stays at step **916** and continues monitoring the new component's traffic and error state. When a success occurs, step **920** occurs. At step **920**, all traffic is migrated to the new component. At step **922**, a shutdown of old instances on the old component can occur. At step **924**, a new gateway and load balancer can be launched.

[0088] FIG. **10** is a flow diagram illustrating an embodiment of a cloud migration process. In one example, the tool can be embodied in a computer system, the system including at least one hardware processor and at least one non-transitory memory storing instructions, which, when executed by the at least one hardware processor, cause the system to perform the process **1000**.

[0089] At step **1002**, the system can receive data, including a specification for a cloud solution. The cloud solution includes a computing service implemented on the cloud infrastructure of a cloud service provider. The specification for the cloud solution indicates a type of cloud solution and at least one requirement of the cloud solution. The type of cloud solution includes infrastructure-as-a-service, platform-as-a-service, software-as-a-service, or function-as-a-service. In one example, the at least one requirement

includes an ability of the cloud infrastructure to support a threshold user traffic volume or a user traffic type, provisioning speed of the cloud infrastructure, reliability of the cloud infrastructure, security protocols, technology, or controls enacted by the cloud service provider, scalability of the cloud solution on the cloud infrastructure where scalability of the cloud solution is the ability to increase the amount of computing power, storage, or bandwidth provided to the cloud solution, or cost to host the cloud service on the cloud infrastructure of the cloud service provider.

[0090] At step **1004**, the system can determine a list of cloud service providers having a cloud infrastructure that is capable of hosting the cloud solution and an ability to satisfy the at least one requirement of the cloud solution. The cloud infrastructure includes hardware components and software that the cloud solution is hosted on. In one example, the system lowers an amount of greenhouse gas emissions by prioritizing the deployment of the cloud solution on high-efficiency hardware components of the cloud infrastructure, which reduces an amount of energy consumed to host the cloud solution on the cloud infrastructure.

[0091] At step **1006**, the system can rank the cloud service providers based on the capability to host the cloud solution and the ability to satisfy the at least one requirement of the cloud solution. In one example, the system includes a user interface. The system can calculate a metric score for each cloud service provider on the list of cloud service providers. The metric score is a calculation that converts the at least one requirement to a single number. The system can cause display of the metric score and can cause display of the ranking of cloud service providers.

[0092] At step **1008**, the system can select the cloud service provider based on the ranking of cloud service providers, where the cloud service provider selected is ranked above a threshold ranking.

[0093] At step **1010**, the system can generate an implementation procedure based on an Application Programming Interface (API) specification of the cloud service provider, where the implementation procedure includes a process of deploying the cloud solution on the cloud infrastructure of the cloud service provider. The API specification determines a requirement for the cloud solution to be deployed on the cloud infrastructure. In one example, the implementation procedure can cause the system to generate, using the LLM, a common cloud source code, where the common cloud source code is a series of instructions defining the cloud solution. The system can convert, using the LLM, the common cloud source code to the programming syntax required for the API of the cloud service provider. Each cloud service provider uses a unique programming syntax, and converting the common cloud source code to the programming syntax of the cloud service provider allows the cloud solution to be deployed on the cloud infrastructure of the cloud service provider. The system can deploy, using the programming syntax of the cloud service provider, the cloud solution on the cloud infrastructure of the cloud service provider.

[0094] At step **1012**, the system can deploy, using a large language model (LLM), the cloud solution on the cloud infrastructure of the cloud service provider in accordance with the implementation procedure where the LLM generates a programming syntax required for the API.

[0095] In one example, where the cloud solution is migrated from a current cloud service provider to a new cloud service provider, the system can search for the new cloud service provider. The system can rank the new cloud service provider against the current cloud service provider, where the new cloud service provider ranks above the current cloud service provider. The system can deploy the cloud solution on the cloud infrastructure of the new cloud service provider. Deploying the cloud solution includes removing the cloud solution from the cloud infrastructure of the current cloud service provider and where no disruption to the cloud solution occurs during the deployment of the cloud solution on the cloud infrastructure of the new cloud service provider.

[0096] In one example, where the cloud solution is deployed on the cloud infrastructure of a current cloud service provider, the system can determine an updated list of cloud service providers. The system can rank the updated list of cloud service providers against the current cloud service provider. The current cloud service provider ranks above the updated list of cloud service providers. The system can evaluate the cloud service provider based on the at least one requirement. The evaluation determines that redeploying the cloud solution increases the ability of the cloud infrastructure of the cloud service provider to satisfy the at least one requirement. The system can determine an updated implementation procedure, where the updated implementation procedure is a process of redeploying the cloud solution on the cloud infrastructure of the cloud service provider. The system can redeploy the cloud solution on the cloud infrastructure of the cloud service provider based on the updated implementation procedure.

Computer System

[0097] FIG. **11** is a block diagram that illustrates an example of a computer system **1100** in which at least some operations described herein can be implemented. As shown, the computer system **1100** can include: one or more processors **1102**, main memory **1106**, non-volatile memory **1110**, a network interface device **1112**, a video display device **1118**, an input/output device **1120**, a control device **1122** (e.g., keyboard and pointing device), a drive unit **1124** that includes a machine-readable (storage) medium **1126**, and a signal generation device **1130** that are communicatively connected to a bus **1116**. The bus **1116** represents one or more physical buses and/or point-to-point connections that are connected by appropriate bridges, adapters, or controllers. Various common components (e.g., cache memory) are omitted from FIG. **11** for brevity. Instead, the computer system **1100** is intended to illustrate a hardware device on which components illustrated or described relative to the examples of the figures and any other components described in this specification can be implemented.

[0098] The computer system **1100** can take any suitable physical form. For example, the computing system **1100** can share a similar architecture as that of a server computer, personal computer (PC), tablet computer, mobile telephone, game console, music player, wearable electronic device, network-connected ("smart") device (e.g., a television or home assistant device), AR/VR systems (e.g., head-mounted display), or any electronic device capable of executing a set of instructions that specify action(s) to be taken by the computing system **1100**. In some implementations, the computer system **1100** can be an embedded computer system, a system-on-chip (SOC), a single-board computer system (SBC), or a distributed system such as a mesh of computer

systems, or it can include one or more cloud components in one or more networks. Where appropriate, one or more computer systems **1100** can perform operations in real time, in near real time, or in batch mode.

[0099] The network interface device **1112** enables the computing system **1100** to mediate data in a network **1114** with an entity that is external to the computing system **1100** through any communication protocol supported by the computing system **1100** and the external entity. Examples of the network interface device **1112** include a network adapter card, a wireless network interface card, a router, an access point, a wireless router, a switch, a multilayer switch, a protocol converter, a gateway, a bridge, a bridge router, a hub, a digital media receiver, and/or a repeater, as well as all wireless elements noted herein.

[0100] The memory (e.g., main memory **1106**, non-volatile memory **1110**, machine-readable medium **1126**) can be local, remote, or distributed. Although shown as a single medium, the machine-readable medium **1126** can include multiple media (e.g., a centralized/distributed database and/or associated caches and servers) that store one or more sets of instructions **1128**. The machine-readable medium **1126** can include any medium that is capable of storing, encoding, or carrying a set of instructions for execution by the computing system **1100**. The machine-readable medium **1126** can be non-transitory or comprise a non-transitory device. In this context, a non-transitory storage medium can include a device that is tangible, meaning that the device has a concrete physical form, although the device can change its physical state. Thus, for example, non-transitory refers to a device remaining tangible despite this change in state.

[0101] Although implementations have been described in the context of fully functioning computing devices, the various examples are capable of being distributed as a program product in a variety of forms. Examples of machine-readable storage media, machine-readable media, or computer-readable media include recordable-type media such as volatile and non-volatile memory **1110**, removable flash memory, hard disk drives, optical disks, and transmission-type media such as digital and analog communication links.

[0102] In general, the routines executed to implement examples herein can be implemented as part of an operating system or a specific application, component, program, object, module, or sequence of instructions (collectively referred to as "computer programs"). The computer programs typically comprise one or more instructions (e.g., instructions **1104**, **1108**, **1128**) set at various times in various memory and storage devices in computing device(s). When read and executed by the processor **1102**, the instruction(s) cause the computing system **1100** to perform operations to execute elements involving the various aspects of the disclosure.

Remarks

[0103] The terms "example," "embodiment," and "implementation" are used interchangeably. For example, references to "one example" or "an example" in the disclosure can be, but not necessarily are, references to the same implementation; and such references mean at least one of the implementations. The appearances of the phrase "in one example" are not necessarily all referring to the same example, nor are separate or alternative examples mutually exclusive of other examples. A feature, structure, or characteristic described in connection with an example can be included in another example of the disclosure. Moreover, various features are described that can be exhibited by some examples and not by others. Similarly, various requirements are described that can be requirements for some examples but not for other examples.

[0104] The terminology used herein should be interpreted in its broadest reasonable manner, even though it is being used in conjunction with certain specific examples of the invention. The terms used in the disclosure generally have their ordinary meanings in the relevant technical art, within the context of the disclosure, and in the specific context where each term is used. A recital of alternative language or synonyms does not exclude the use of other synonyms. Special significance should not be placed upon whether or not a term is elaborated or discussed herein. The use of highlighting has no influence on the scope and meaning of a term. Further, it will be appreciated that the same thing can be said in more than one way.

[0105] Unless the context clearly requires otherwise, throughout the description and the claims, the words "comprise," "comprising," and the like are to be construed in an inclusive sense, as opposed to an exclusive or exhaustive sense—that is to say, in the sense of "including, but not limited to." As used herein, the terms "connected," "coupled," and any variants thereof mean any connection or coupling, either direct or indirect, between two or more elements; the coupling or connection between the elements can be physical, logical, or a combination thereof. Additionally, the words "herein," "above," "below," and words of similar import can refer to this application as a whole and not to any particular portions of this application. Where context permits, words in the above Detailed Description using the singular or plural number may also include the plural or singular number, respectively. The word "or" in reference to a list of two or more items covers all of the following interpretations of the word: any of the items in the list, all of the items in the list, and any combination of the items in the list. The term "module" refers broadly to software components, firmware components, and/or hardware components.

[0106] While specific examples of technology are described above for illustrative purposes, various equivalent modifications are possible within the scope of the invention, as those skilled in the relevant art will recognize. For example, while processes or blocks are presented in a given order, alternative implementations can perform routines having steps, or employ systems having blocks, in a different order, and some processes or blocks may be deleted, moved, added, subdivided, combined, and/or modified to provide alternative or sub-combinations. Each of these processes or blocks can be implemented in a variety of different ways. Also, while processes or blocks are at times shown as being performed in series, these processes or blocks can instead be performed or implemented in parallel, or can be performed at different times. Further, any specific numbers noted herein are only examples such that alternative implementations can employ differing values or ranges.

[0107] Details of the disclosed implementations can vary considerably in specific implementations while still being encompassed by the disclosed teachings. As noted above, particular terminology used when describing features or aspects of the invention should not be taken to imply that the terminology is being redefined herein to be restricted to any

specific characteristics, features, or aspects of the invention with which that terminology is associated. In general, the terms used in the following claims should not be construed to limit the invention to the specific examples disclosed herein, unless the above Detailed Description explicitly defines such terms. Accordingly, the actual scope of the invention encompasses not only the disclosed examples but also all equivalent ways of practicing or implementing the invention under the claims. Some alternative implementations can include additional elements to those implementations described above or include fewer elements.

[0108] Any patents and applications and other references noted above, and any that may be listed in accompanying filing papers, are incorporated herein by reference in their entireties, except for any subject matter disclaimers or disavowals, and except to the extent that the incorporated material is inconsistent with the express disclosure herein, in which case the language in this disclosure controls. Aspects of the invention can be modified to employ the systems, functions, and concepts of the various references described above to provide yet further implementations of the invention.

[0109] To reduce the number of claims, certain implementations are presented below in certain claim forms, but the applicant contemplates various aspects of an invention in other forms. For example, aspects of a claim can be recited in a means-plus-function form or in other forms, such as being embodied in a computer-readable medium. A claim intended to be interpreted as a means-plus-function claim will use the words "means for." However, the use of the term "for" in any other context is not intended to invoke a similar interpretation. The applicant reserves the right to pursue such additional claim forms either in this application or in a continuing application.

1. A system comprising:

at least one hardware processor; and

at least one non-transitory memory storing instructions, which, when executed by the at least one hardware processor, cause the system to:

receive data including a specification for a cloud solution,

wherein the cloud solution includes a computing service implemented on a cloud infrastructure of a cloud service provider,

wherein the specification for the cloud solution indicates a type of cloud solution and at least one requirement of the cloud solution,

wherein the at least one requirement of the cloud solution includes a threshold provisioning speed of the cloud infrastructure and at least one of the following: one or more security protocols enacted by the cloud service provider or an ability of the cloud infrastructure to support a threshold number of concurrent users, and

wherein the type of cloud solution includes infrastructure-as-a-service, platform-as-a-service, software-as-a-service, or function as-a-service;

determine a list of cloud service providers having a cloud infrastructure that is capable of hosting the cloud solution and an ability to satisfy the at least one requirement of the cloud solution,

wherein the cloud infrastructure includes hardware components and software that the cloud solution is hosted on;

assign a weighted value to the at least one requirement of the cloud solution;

calculate a metric score for each of the cloud service providers based on the weighted value of the at least one requirement of the cloud solution,

wherein the metric score indicates a capability of a cloud service providers to host the cloud solution;

rank the cloud service providers based on the metric score;

select the cloud service provider based on the ranking of cloud service providers,

wherein the cloud service provider selected is ranked above a threshold ranking;

generate an implementation procedure based on an Application Programming Interface (API) specification of the cloud service provider,

wherein the implementation procedure includes a process of deploying the cloud solution on the cloud infrastructure of the cloud service provider, and

wherein the API specification determines a requirement for the cloud solution to be deployed on the cloud infrastructure; and

deploy, using a large language model (LLM), the cloud solution on the cloud infrastructure of the cloud service provider in accordance with the implementation procedure,

wherein the LLM generates a programming syntax required for the API.

2. The system of claim **1**, wherein the cloud solution is migrated from a current cloud service provider to a new cloud service provider, the instructions further cause the system to:

search for the new cloud service provider;

rank the new cloud service provider against the current cloud service provider,

wherein the new cloud service provider ranks above the current cloud service provider; and

deploy the cloud solution on the cloud infrastructure of the new cloud service provider,

wherein deploying the cloud solution includes removing the cloud solution from the cloud infrastructure of the current cloud service provider, and

wherein no disruption to the cloud solution occurs during the deployment of the cloud solution on the cloud infrastructure of the new cloud service provider.

3. The system of claim **1**, wherein the implementation procedure further causes the system to:

generate, using the LLM, a common cloud source code,

wherein the common cloud source code is a series of instructions defining the cloud solution;

convert, using the LLM, the common cloud source code to the programming syntax required for the API of the cloud service provider,

wherein each cloud service provider uses a unique programming syntax, and

wherein converting the common cloud source code to the programming syntax of the cloud service provider allows the cloud solution to be deployed on the cloud infrastructure of the cloud service provider; and

deploy, using the programming syntax of the cloud service provider, the cloud solution on the cloud infrastructure of the cloud service provider.

4. The system of claim **1**, wherein the at least one requirement includes:

an ability of the cloud infrastructure to support a threshold user traffic volume or a user traffic type;

reliability of the cloud infrastructure;

security technology or controls enacted by the cloud service provider;

scalability of the cloud solution on the cloud infrastructure,

wherein scalability of the cloud solution is the ability to increase an amount of computing power, storage, or bandwidth provided to the cloud solution; or

cost to host the cloud solution on the cloud infrastructure of the cloud service provider.

5. The system of claim **1**, wherein the cloud solution is deployed on the cloud infrastructure of a current cloud service provider, the instructions further cause the system to:

determine an updated list of cloud service providers;

rank the updated list of cloud service providers against the current cloud service provider,

wherein the current cloud service provider ranks above the updated list of cloud service providers;

evaluate the cloud service provider based on the at least one requirement,

wherein the evaluation determines that redeploying the cloud solution increases the ability of the cloud infrastructure of the cloud service provider to satisfy the at least one requirement;

determine an updated implementation procedure,

wherein the updated implementation procedure is a process of redeploying the cloud solution on the cloud infrastructure of the cloud service provider; and

redeploy the cloud solution on the cloud infrastructure of the cloud service provider based on the updated implementation procedure.

6. The system of claim **1**, wherein the system includes a user interface, the system being further caused to:

calculate a metric score for each cloud service provider on the list of cloud service providers,

wherein the metric score is a calculation that converts the at least one requirement to a single number;

cause display of the metric score; and

cause display of the ranking of cloud service providers.

7. The system of claim **1**, wherein the system lowers an amount of greenhouse gas emissions by prioritizing the deployment of the cloud solution on high-efficiency hardware components of the cloud infrastructure, which reduces an amount of energy consumed to host the cloud solution on the cloud infrastructure.

8. A non-transitory, computer-readable storage medium comprising instructions recorded thereon, wherein the instructions, when executed by at least one data processor of a system, cause the system to:

receive data including a specification for a cloud solution,

wherein the cloud solution includes a computing service implemented on a cloud infrastructure of a cloud service provider,

wherein the specification for the cloud solution indicates a type of cloud solution and at least one requirement of the cloud solution,

wherein the at least one requirement of the cloud solution includes a provisioning speed of the cloud infrastructure and at least one of the following: one or more security protocols enacted by the cloud service provider or an ability of the cloud infrastructure to support a threshold number of concurrent users, and

wherein the type of cloud solution includes infrastructure-as-a-service, platform-as-a-service, software-as-a-service, or function-as-a-service;

determine a list of cloud service providers having a cloud infrastructure that is capable of hosting the cloud solution and an ability to satisfy the at least one requirement of the cloud solution,

wherein the cloud infrastructure includes hardware components and software that the cloud solution is hosted on;

assign a weighted value to the at least one requirement of the cloud solution;

calculate a metric score for each of the cloud service providers based on the weighted value of the at least one requirement of the cloud solution,

wherein the metric score indicates a capability of a cloud service providers to host the cloud solution;

rank the cloud service providers based on the metric score;

select the cloud service provider based on the ranking of cloud service providers,

wherein the cloud service provider selected is ranked above a threshold ranking;

generate an implementation procedure based on an Application Programming Interface (API) specification of the cloud service provider,

wherein the implementation procedure includes a process of deploying the cloud solution on the cloud infrastructure of the cloud service provider, and

wherein the API specification determines a requirement for the cloud solution to be deployed on the cloud infrastructure; and

deploy, using a large language model (LLM), the cloud solution on the cloud infrastructure of the cloud service provider in accordance with the implementation procedure,

wherein the LLM generates a programming syntax required for the API.

9. The non-transitory, computer-readable storage medium of claim **8**, wherein the cloud solution is migrated from a current cloud service provider to a new cloud service provider, the instructions further cause the system to:

search for the new cloud service provider;

rank the new cloud service provider against the current cloud service provider,

wherein the new cloud service provider ranks above the current cloud service provider; and

deploy the cloud solution on the cloud infrastructure of the new cloud service provider,

wherein deploying the cloud solution includes removing the cloud solution from the cloud infrastructure of the current cloud service provider, and

wherein no disruption to the cloud solution occurs during the deployment of the cloud solution on the cloud infrastructure of the new cloud service provider.

10. The non-transitory, computer-readable storage medium of claim 8, wherein the implementation procedure further causes the system to:
- generate, using the LLM, a common cloud source code,
  - wherein the common cloud source code is a series of instructions defining the cloud solution;
- convert, using the LLM, the common cloud source code to the programming syntax required for the API of the cloud service provider,
  - wherein each cloud service provider uses a unique programming syntax, and
  - wherein converting the common cloud source code to the programming syntax of the cloud service provider allows the cloud solution to be deployed on the cloud infrastructure of the cloud service provider; and
- deploy, using the programming syntax of the cloud service provider, the cloud solution on the cloud infrastructure of the cloud service provider.

11. The non-transitory, computer-readable storage medium of claim 8, wherein the at least one requirement includes:
- an ability of the cloud infrastructure to support a threshold user traffic volume or a user traffic type;
- reliability of the cloud infrastructure;
- security technology or controls enacted by the cloud service provider;
- scalability of the cloud solution on the cloud infrastructure,
  - wherein scalability of the cloud solution is the ability to increase an amount of computing power, storage, or bandwidth provided to the cloud solution; or
- cost to host the cloud solution on the cloud infrastructure of the cloud service provider.

12. The non-transitory, computer-readable storage medium of claim 8, wherein the cloud solution is deployed on the cloud infrastructure of a current cloud service provider, the instructions further cause the system to:
- determine an updated list of cloud service providers;
- rank the updated list of cloud service providers against the current cloud service provider,
  - wherein the current cloud service provider ranks above the updated list of cloud service providers;
- evaluate the cloud service provider based on the at least one requirement,
  - wherein the evaluation determines that redeploying the cloud solution increases the ability of the cloud infrastructure of the cloud service provider to satisfy the at least one requirement;
- determine an updated implementation procedure,
  - wherein the updated implementation procedure is a process of redeploying the cloud solution on the cloud infrastructure of the cloud service provider; and
- redeploy the cloud solution on the cloud infrastructure of the cloud service provider based on the updated implementation procedure.

13. The non-transitory, computer-readable storage medium of claim 8, wherein the system includes a user interface, the system being further caused to:
- calculate a metric score for each cloud service provider on the list of cloud service providers,
  - wherein the metric score is a calculation that converts the at least one requirement to a single number;
- cause display of the metric score; and
- cause display of the ranking of cloud service providers.

14. The non-transitory, computer-readable storage medium of claim 8, wherein the system lowers an amount of greenhouse gas emissions by prioritizing the deployment of the cloud solution on high-efficiency hardware components of the cloud infrastructure, which reduces an amount of energy consumed to host the cloud solution on the cloud infrastructure.

15. A method comprising:
- receiving data including a specification for a cloud solution,
  - wherein the cloud solution includes a computing service implemented on a cloud infrastructure of a cloud service provider,
  - wherein the specification for the cloud solution indicates a type of cloud solution and at least one requirement of the cloud solution, and
  - wherein the at least one requirement of the cloud solution includes a provisioning speed of the cloud infrastructure and at least one of the following: one or more security protocols enacted by the cloud service provider or an ability of the cloud infrastructure to support a threshold number of concurrent users;
- determining a list of cloud service providers having a cloud infrastructure that is capable of hosting the cloud solution and an ability to satisfy the at least one requirement of the cloud solution;
- assigning a weighted value to the at least one requirement of the cloud solution;
- calculating a metric score for each of the cloud service providers based on the weighted value of the at least one requirement of the cloud solution,
  - wherein the metric score indicates a capability of a cloud service providers to host the cloud solution;
- ranking the cloud service providers based on the metric score;
- selecting the cloud service provider based on the ranking of cloud service providers;
- generating an implementation procedure based on an Application Programming Interface (API) specification of the cloud service provider,
  - wherein the implementation procedure includes a process of deploying the cloud solution on the cloud infrastructure of the cloud service provider, and
  - wherein the API specification determines a requirement for the cloud solution to be deployed on the cloud infrastructure; and
- deploying, using a large language model (LLM), the cloud solution on the cloud infrastructure of the cloud service provider in accordance with the implementation procedure,
  - wherein the LLM generates a programming syntax required for the API.

16. The method of claim 15, wherein the cloud solution is migrated from a current cloud service provider to a new cloud service provider, further comprising:
- searching for the new cloud service provider;
- ranking the new cloud service provider against the current cloud service provider,
  - wherein the new cloud service provider ranks above the current cloud service provider; and

deploying the cloud solution on the cloud infrastructure of the new cloud service provider,

wherein deploying the cloud solution includes removing the cloud solution from the cloud infrastructure of the current cloud service provider, and

wherein no disruption to the cloud solution occurs during the deployment of the cloud solution on the cloud infrastructure of the new cloud service provider.

17. The method of claim **15**, further comprising:

generating, using the LLM, a common cloud source code,

wherein the common cloud source code is a series of instructions defining the cloud solution;

converting, using the LLM, the common cloud source code to the programming syntax required for the API of the cloud service provider,

wherein each cloud service provider uses a unique programming syntax, and

wherein converting the common cloud source code to the programming syntax of the cloud service provider allows the cloud solution to be deployed on the cloud infrastructure of the cloud service provider; and

deploying, using the programming syntax of the cloud service provider, the cloud solution on the cloud infrastructure of the cloud service provider.

18. The method of claim **15**, wherein the at least one requirement includes:

an ability of the cloud infrastructure to support a threshold user traffic volume or a user traffic type;

reliability of the cloud infrastructure;

security technology or controls enacted by the cloud service provider;

scalability of the cloud solution on the cloud infrastructure,

wherein scalability of the cloud solution is the ability to increase an amount of computing power, storage, or bandwidth provided to the cloud solution; or

cost to host the cloud solution on the cloud infrastructure of the cloud service provider.

19. The method of claim **15**, wherein the cloud solution is deployed on the cloud infrastructure of a current cloud service provider, further comprising:

determining an updated list of cloud service providers;

ranking the updated list of cloud service providers against the current cloud service provider,

wherein the current cloud service provider ranks above the updated list of cloud service providers;

evaluating the cloud service provider based on the at least one requirement,

wherein the evaluation determines that redeploying the cloud solution increases the ability of the cloud infrastructure of the cloud service provider to satisfy the at least one requirement;

determining an updated implementation procedure,

wherein the updated implementation procedure is a process of redeploying the cloud solution on the cloud infrastructure of the cloud service provider; and

redeploying the cloud solution on the cloud infrastructure of the cloud service provider based on the updated implementation procedure.

20. The method of claim **15**, further comprising:

calculating a metric score for each cloud service provider on the list of cloud service providers,

wherein the metric score is a calculation that converts the at least one requirement to a single number;

causing display of the metric score on a user interface; and

causing display of the ranking of cloud service providers on the user interface.

* * * * *