US 20240296071A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2024/0296071 A1

TOAL et al. (43) Pub. Date: Sep. 5, 2024

(54) **AUTOMATICALLY IDENTIFYING AND RIGHT SIZING INSTANCES**

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

(72) Inventors: **Brian TOAL**, San Francisco, CA (US); **Manpreet SINGH**, Hyderabad (IN)

(21) Appl. No.: **18/657,187**

(22) Filed: **May 7, 2024**

Related U.S. Application Data

(63) Continuation of application No. 17/854,652, filed on Jun. 30, 2022, now Pat. No. 12,008,407, which is a continuation of application No. 16/566,209, filed on Sep. 10, 2019, now Pat. No. 11,379,266.

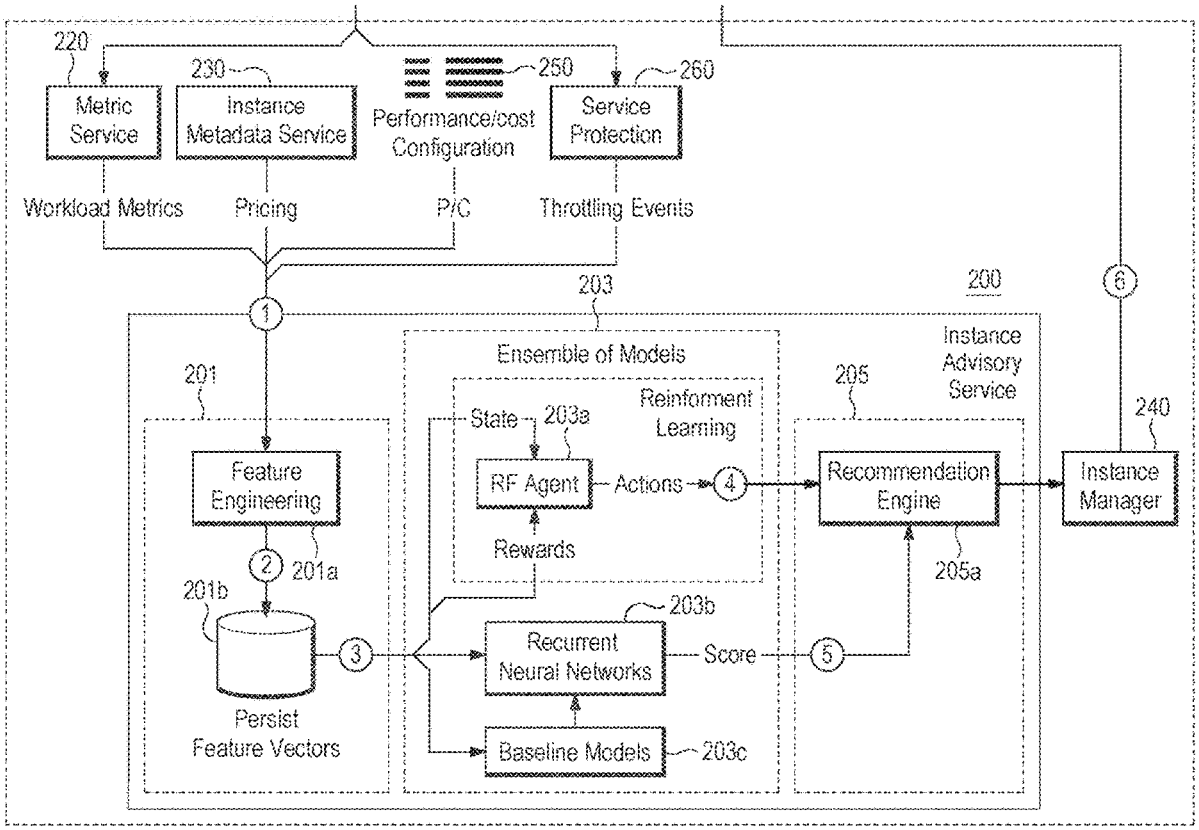
Publication Classification

(51) **Int. Cl.**
G06F 9/50 (2006.01)

(52) **U.S. Cl.**
CPC ***G06F 9/5027*** (2013.01); ***G06F 9/5083*** (2013.01); *G06F 2209/501* (2013.01); *G06F 2209/5011* (2013.01); *G06F 2209/505* (2013.01)

(57) **ABSTRACT**

A system is disclosed. The system includes a resource monitor to monitor a resource utilization of a set of resources of one or more instances, the resource utilization corresponding to a first level of performance and cost and an instance type determiner to, based on the resource utilization, determine if there is an instance type for at least one of the one or more instances, with a resource profile, that will provide a second level of performance and cost that is closer to a default level of performance and cost than the first level of performance and cost. In addition, the system also includes an instance type recommender to, based on the determining, perform one of making and not making a recommendation to replace the instance type of the at least one of the one or more instances.



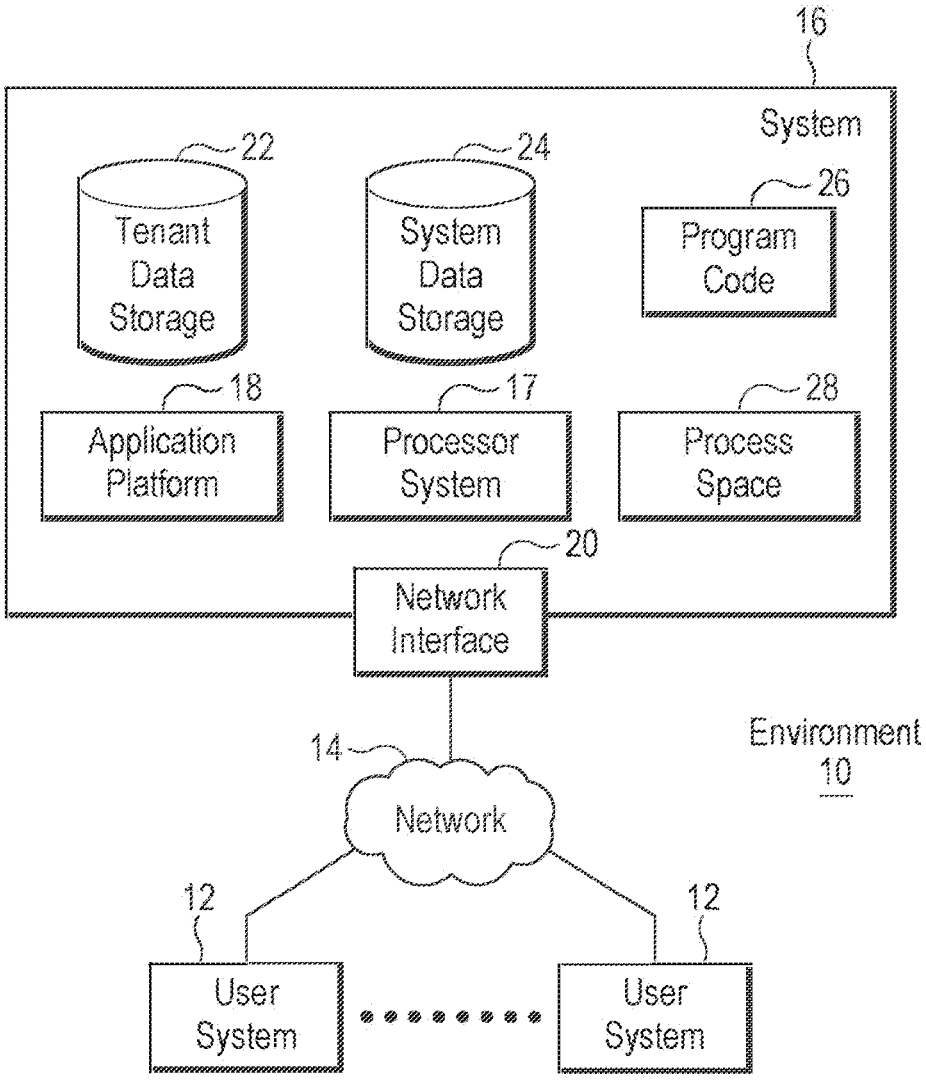

16
System
22
Tenant Data Storage
24
System Data Storage
26
Program Code
18
Application Platform
17
Processor System
28
Process Space
20
Network Interface
Environment 10
14
Network
12
12
User System
User System

FIG. 1A

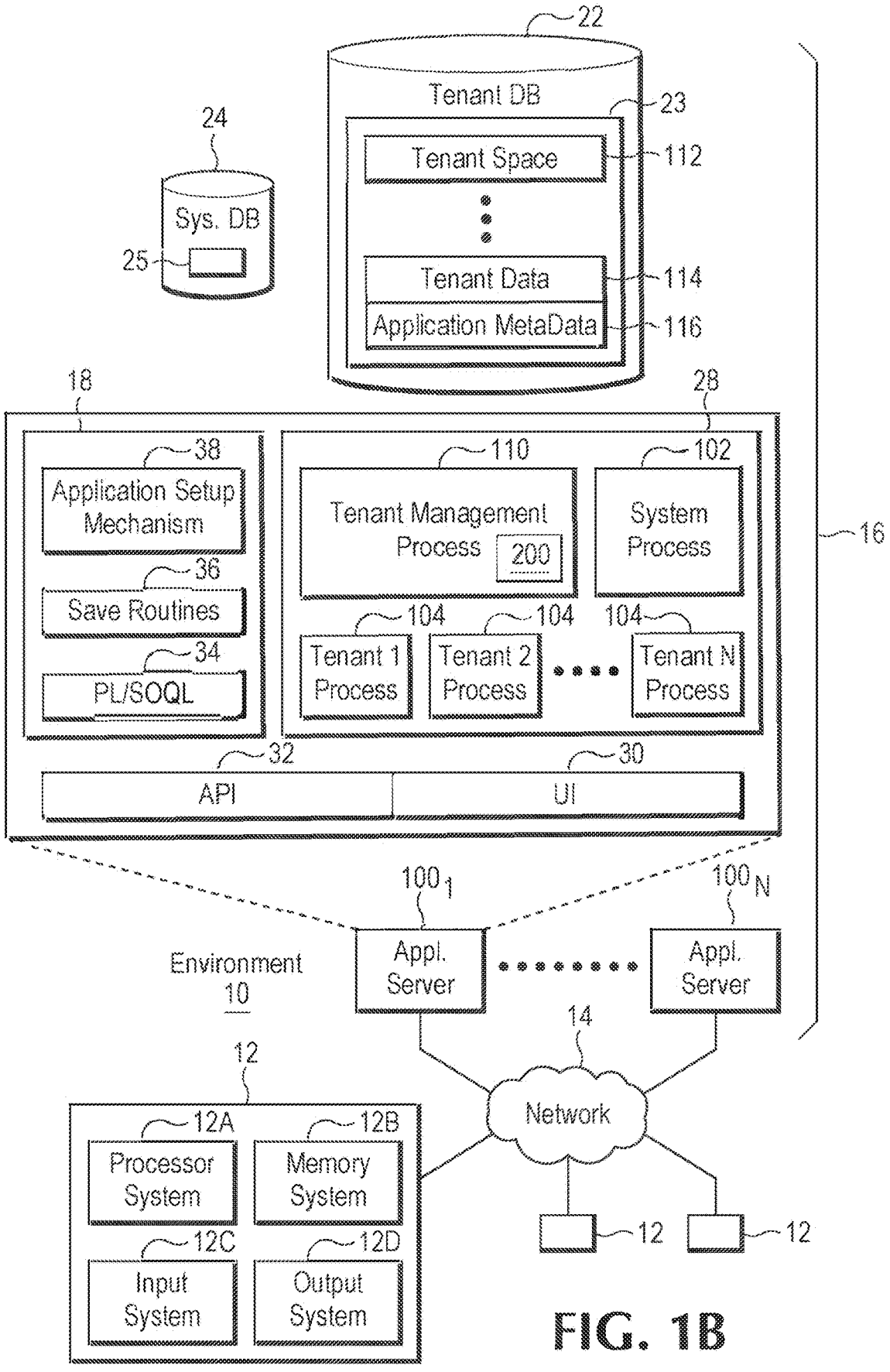

22
Tenant DB
23
24
Tenant Space
112
Sys. DB
25
Tenant Data
114
Application MetaData
116
18
28
38
110
102
Application Setup Mechanism
Tenant Management Process
200
System Process
16
36
Save Routines
104
104
104
34
Tenant 1 Process
Tenant 2 Process
Tenant N Process
PL/SOQL
32
30
API
UI
100 1
100 N
Environment 10
Appl. Server
Appl. Server
14
12
Network
12A
12B
Processor System
Memory System
12C
12D
Input System
Output System
12
12

FIG. 1B

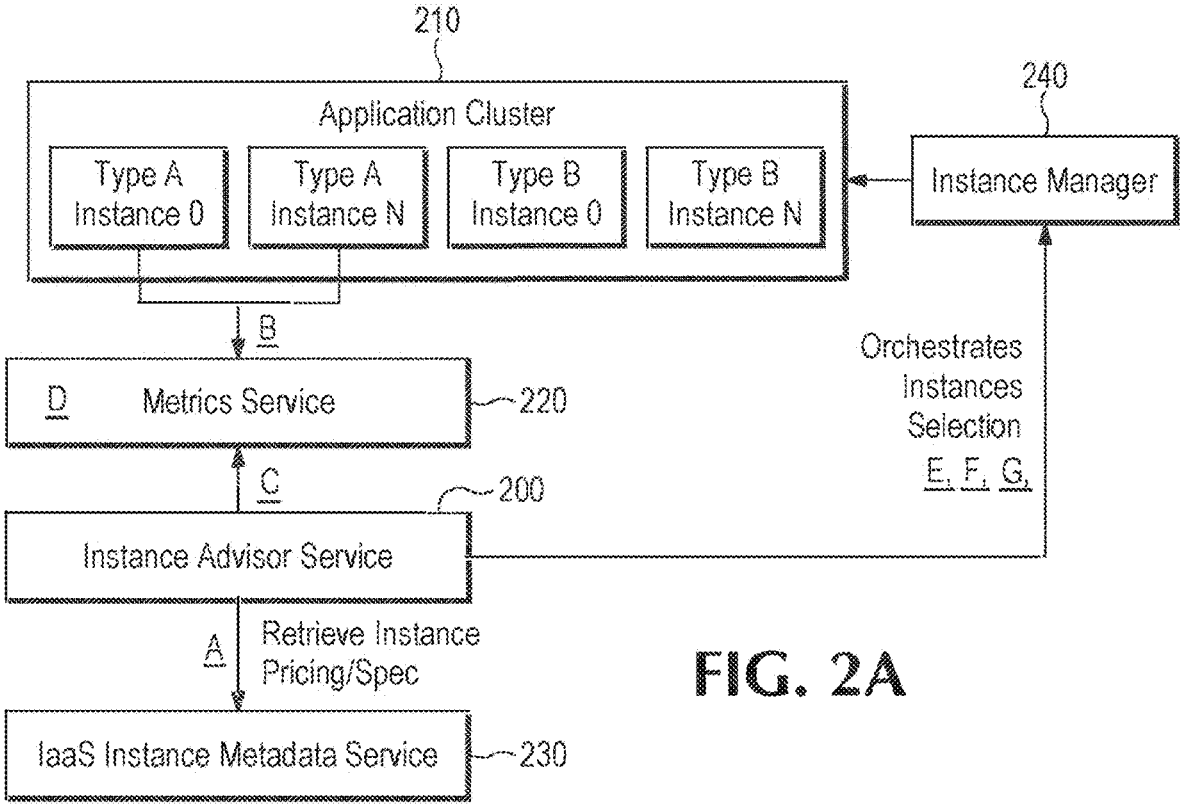


FIG. 2A

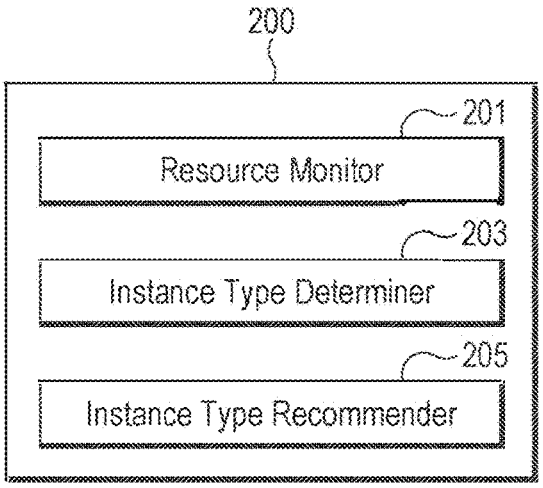


FIG. 2B

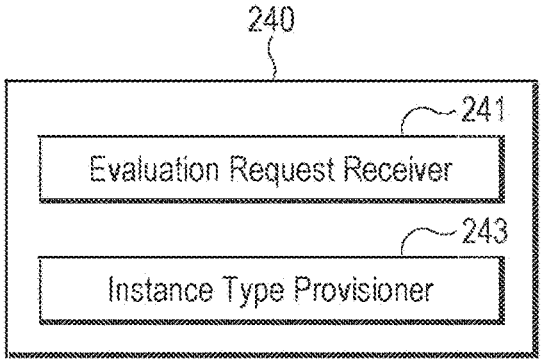


FIG. 2C

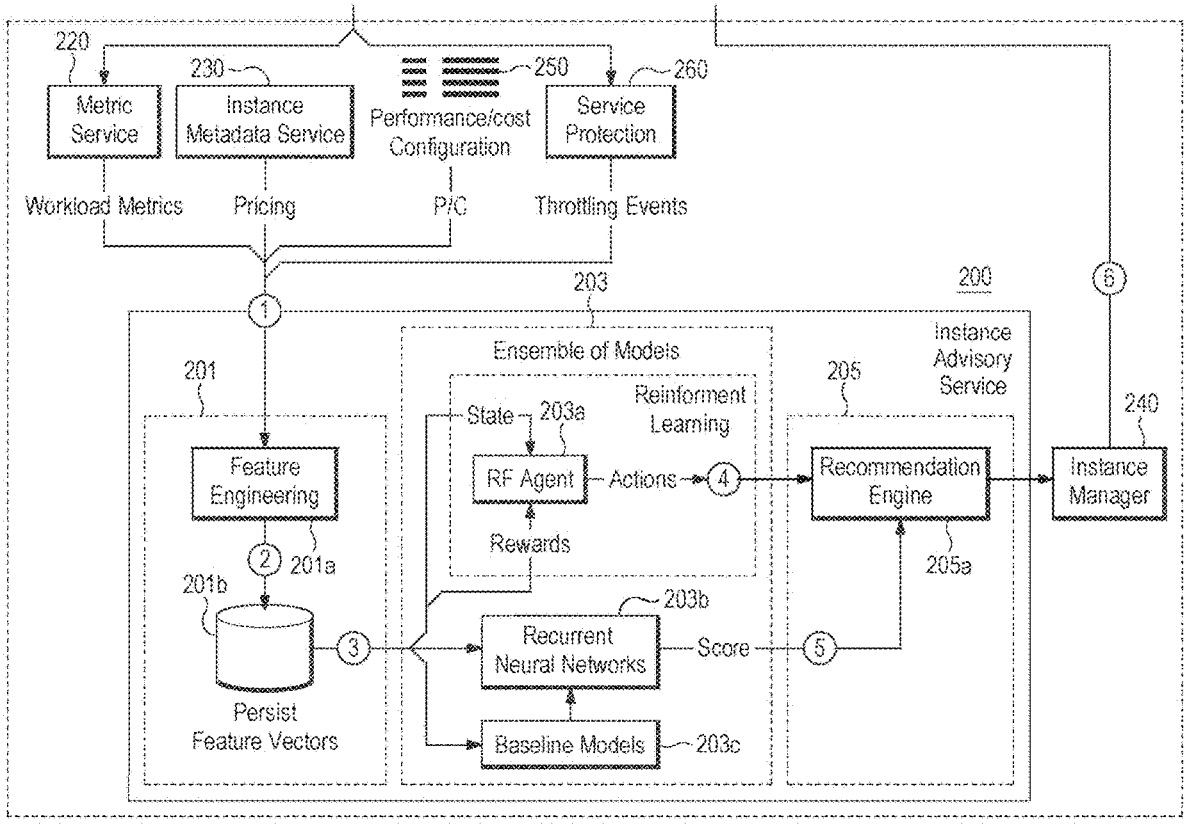

220
230
250
260
Metric Service
Instance Metadata Service
Performance/cost Configuration
Service Protection
Workload Metrics
Pricing
P/C
Throttling Events
203
200
6
1
Ensemble of Models
Instance Advisory Service
201
205
240
State
203a
Reinforment Learning
Feature Engineering
RF Agent
Actions
4
Recommendation Engine
Instance Manager
Rewards
2
201a
205a
201b
203b
Recurrent Neural Networks
Score
3
5
Persist Feature Vectors
Baseline Models
203c

FIG. 2D

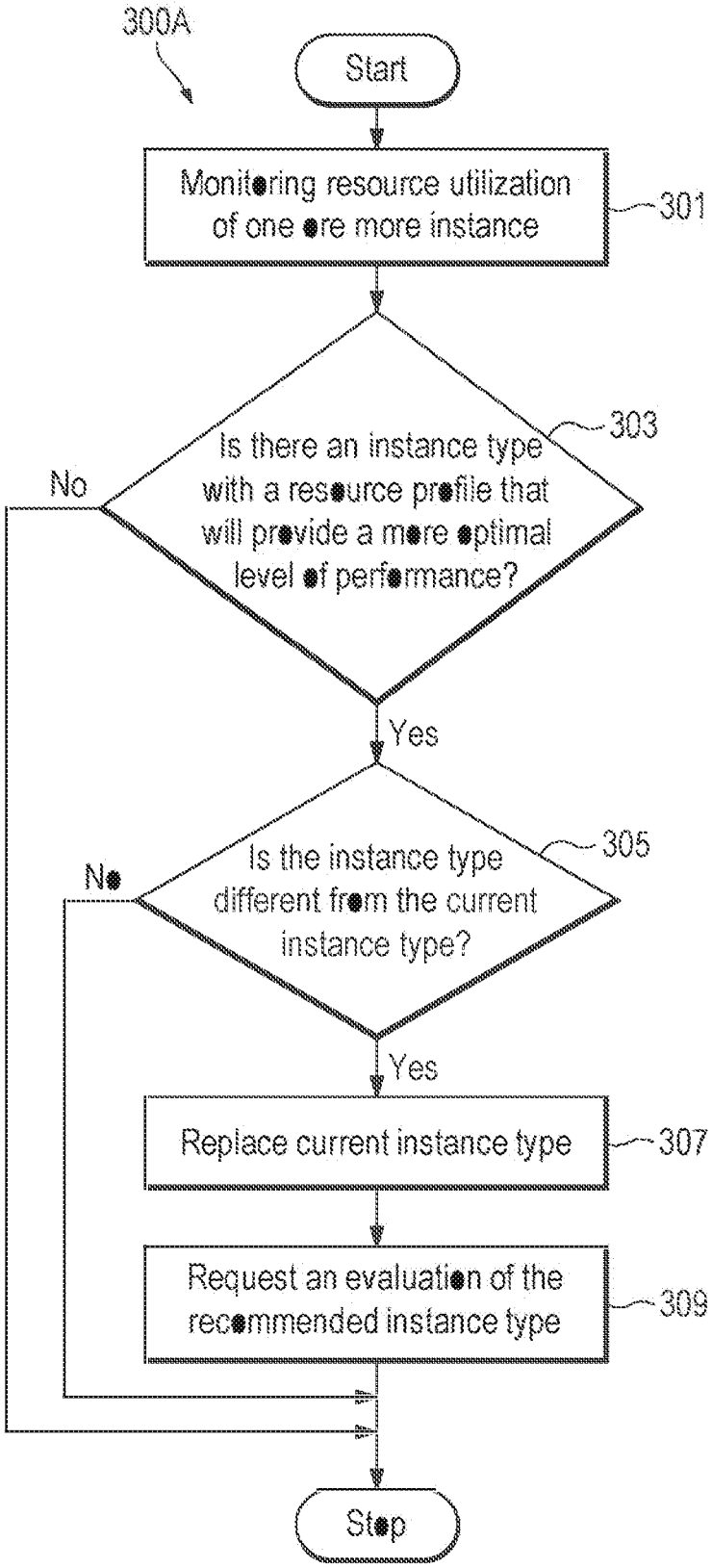

300A
Start
Monitoring resource utilization of one or more instance
301
Is there an instance type with a resource profile that will provide a more optimal level of performance?
303
No
Yes
Is the instance type different from the current instance type?
305
No
Yes
Replace current instance type
307
Request an evaluation of the recommended instance type
309
Stop

FIG. 3A

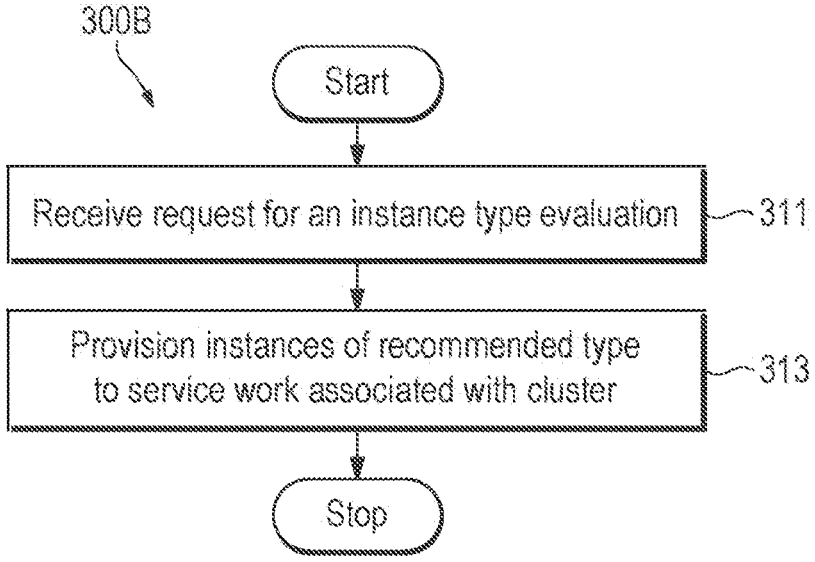

300B
Start
Receive request for an instance type evaluation
311
Provision instances of recommended type to service work associated with cluster
313
Stop

FIG. 3B

AUTOMATICALLY IDENTIFYING AND RIGHT SIZING INSTANCES

COPYRIGHT NOTICE

TECHNICAL FIELD

[0002] One or more implementations relate generally to container instances, and more specifically to automatically identifying and right sizing container instances.

BACKGROUND

[0003] The material discussed in this background section should not be assumed to be prior art merely as a result of its mention in the background section. Similarly, a problem mentioned in the background section or associated with the subject matter of the background section should not be assumed to have been previously recognized in the prior art. The subject matter in the background section merely represents different approaches, which in and of themselves may also correspond to implementations of the claimed technology.

[0004] Infrastructure as a service (IaaS) is a form of cloud computing that provides virtualized computing resources over the internet. In an IaaS model, a cloud provider hosts the infrastructure components traditionally present in an on-premises data center, including servers, storage and networking hardware, as well as the virtualization or hypervisor layer.

[0005] The IaaS provider also supplies a range of services to accompany those infrastructure components. The services can be provided by an application that can run across one to many instances of containers. A container is mapped to an instance type.

[0006] The underutilization of instances that are provided by an IaaS service in terms of their CPU/Memory/Storage usage can be problematic from a cost perspective. In particular, when an instance is purchased on a per year basis the full cost of the instance must be paid whether usage of the resources associated with the instance is maximized or not.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] The included drawings are for illustrative purposes and serve to provide examples of possible structures and operations for the disclosed inventive systems, apparatus, methods and computer-readable storage media. These drawings in no way limit any changes in form and detail that may be made by one skilled in the art without departing from the spirit and scope of the disclosed implementations.

[0008] FIG. **1**A shows a block diagram of an example environment in which an on-demand database service can be used according to some implementations.

[0009] FIG. **1**B shows a block diagram of example implementations of elements of FIG. **1**A and example interconnections between these elements according to some implementations.

[0010] FIG. **2**A illustrates the operation of an instance advisory service shown in FIG. **1**B according to an embodiment.

[0011] FIG. **2**B shows components of an instance advisory service according to an embodiment.

[0012] FIG. **2**C shows components of an instance manager according to an embodiment.

[0013] FIG. **2**D illustrates an example implementation of the instance advisory service and its interaction with a metric service, an instance metadata service, a service protection service, a performance cost configuration service and an instance manager.

[0014] FIG. **3**A is a flowchart of a method for automatically identifying and rightsizing instances according to an embodiment.

[0015] FIG. **3**B is a flowchart of a method for evaluating instances according to an embodiment.

DETAILED DESCRIPTION

[0016] Examples of systems, apparatus, computer-readable storage media, and methods according to the disclosed implementations are described in this section. These examples are being provided solely to add context and aid in the understanding of the disclosed implementations. It will thus be apparent to one skilled in the art that the disclosed implementations may be practiced without some or all of the specific details provided. In other instances, certain process or method operations, also referred to herein as "blocks," have not been described in detail in order to avoid unnecessarily obscuring the disclosed implementations. Other implementations and applications also are possible, and as such, the following examples should not be taken as definitive or limiting either in scope or setting.

[0017] In the following detailed description, references are made to the accompanying drawings, which form a part of the description and in which are shown, by way of illustration, specific implementations. Although these disclosed implementations are described in sufficient detail to enable one skilled in the art to practice the implementations, it is to be understood that these examples are not limiting, such that other implementations may be used and changes may be made to the disclosed implementations without departing from their spirit and scope. For example, the blocks of the methods shown and described herein are not necessarily performed in the order indicated in some other implementations. Additionally, in some other implementations, the disclosed methods may include more or fewer blocks than are described. As another example, some blocks described herein as separate blocks may be combined in some other implementations. Conversely, what may be described herein as a single block may be implemented in multiple blocks in some other implementations. Additionally, the conjunction "or" is intended herein in the inclusive sense where appropriate unless otherwise indicated; that is, the phrase "A, B or C" is intended to include the possibilities of "A," "B," "C," "A and B," "B and C," "A and C" and "A, B and C."

I. Example System Overview

[0018] FIG. **1**A shows a block diagram of an example of an environment **10** in which an on-demand database service can be used in accordance with some implementations. The environment **10** includes user systems **12**, a network **14**, a database system **16** (also referred to herein as a "cloud-based

system"), a processor system **17**, an application platform **18**, a network interface **20**, tenant database **22** for storing tenant data **23**, system database **24** for storing system data **25**, program code **26** for implementing various functions of the system **16**, and process space **28** for executing database system processes and tenant-specific processes, such as running applications as part of an application hosting service. In some other implementations, environment **10** may not have all of these components or systems, or may have other components or systems instead of, or in addition to, those listed above.

[0019] In some implementations, the environment **10** is an environment in which an on-demand database service exists. An on-demand database service, such as that which can be implemented using the system **16**, is a service that is made available to users outside of the enterprise(s) that own, maintain or provide access to the system **16**. As described above, such users generally do not need to be concerned with building or maintaining the system **16**. Instead, resources provided by the system **16** may be available for such users' use when the users need services provided by the system **16**; that is, on the demand of the users. Some on-demand database services can store information from one or more tenants into tables of a common database image to form a multi-tenant database system (MTS). The term "multi-tenant database system" can refer to those systems in which various elements of hardware and software of a database system may be shared by one or more customers or tenants. For example, a given application server may simultaneously process requests for a great number of customers, and a given database table may store rows of data such as feed items for a potentially much greater number of customers. A database image can include one or more database objects. A relational database management system (RDBMS) or the equivalent can execute storage and retrieval of information against the database object(s).

[0020] Application platform **18** can be a framework that allows the applications of system **16** to execute, such as the hardware or software infrastructure of the system **16**. In some implementations, the application platform **18** enables the creation, management and execution of one or more applications developed by the provider of the on-demand database service, users accessing the on-demand database service via user systems **12**, or third party application developers accessing the on-demand database service via user systems **12**.

[0021] In some implementations, the system **16** implements a web-based customer relationship management (CRM) system. For example, in some such implementations, the system **16** includes application servers configured to implement and execute CRM software applications as well as provide related data, code, forms, renderable web pages and documents and other information to and from user systems **12** and to store to, and retrieve from, a database system related data, objects, and Web page content. In some MTS implementations, data for multiple tenants may be stored in the same physical database object in tenant database **22**. In some such implementations, tenant data is arranged in the storage medium(s) of tenant database **22** so that data of one tenant is kept logically separate from that of other tenants so that one tenant does not have access to another tenant's data, unless such data is expressly shared. The system **16** also implements applications other than, or in addition to, a CRM application. For example, the system **16** can provide tenant access to multiple hosted (standard and custom) applications, including a CRM application. User (or third party developer) applications, which may or may not include CRM, may be supported by the application platform **18**. The application platform **18** manages the creation and storage of the applications into one or more database objects and the execution of the applications in one or more virtual machines in the process space of the system **16**.

[0022] According to some implementations, each system **16** is configured to provide web pages, forms, applications, data and media content to user (client) systems **12** to support the access by user systems **12** as tenants of system **16**. As such, system **16** provides security mechanisms to keep each tenant's data separate unless the data is shared. If more than one MTS is used, they may be located in close proximity to one another (for example, in a server farm located in a single building or campus), or they may be distributed at locations remote from one another (for example, one or more servers located in city A and one or more servers located in city B). As used herein, each MTS could include one or more logically or physically connected servers distributed locally or across one or more geographic locations. Additionally, the term "server" is meant to refer to a computing device or system, including processing hardware and process space(s), an associated storage medium such as a memory device or database, and, in some instances, a database application (for example, OODBMS or RDBMS) as is well known in the art. It should also be understood that "server system" and "server" are often used interchangeably herein. Similarly, the database objects described herein can be implemented as part of a single database, a distributed database, a collection of distributed databases, a database with redundant online or offline backups or other redundancies, etc., and can include a distributed database or storage network and associated processing intelligence.

[0023] The network **14** can be or include any network or combination of networks of systems or devices that communicate with one another. For example, the network **14** can be or include any one or any combination of a LAN (local area network), WAN (wide area network), telephone network, wireless network, cellular network, point-to-point network, star network, token ring network, hub network, or other appropriate configuration. The network **14** can include a TCP/IP (Transfer Control Protocol and Internet Protocol) network, such as the global internetwork of networks often referred to as the "Internet" (with a capital "I"). The Internet will be used in many of the examples herein. However, it should be understood that the networks that the disclosed implementations can use are not so limited, although TCP/IP is a frequently implemented protocol.

[0024] The user systems **12** can communicate with system **16** using TCP/IP and, at a higher network level, other common Internet protocols to communicate, such as HTTP, FTP, AFS, WAP, etc. In an example where HTTP is used, each user system **12** can include an HTTP client commonly referred to as a "web browser" or simply a "browser" for sending and receiving HTTP signals to and from an HTTP server of the system **16**. Such an HTTP server can be implemented as the sole network interface **20** between the system **16** and the network **14**, but other techniques can be used in addition to or instead of these techniques. In some implementations, the network interface **20** between the system **16** and the network **14** includes load sharing functionality, such as round-robin HTTP request distributors to

balance loads and distribute incoming HTTP requests evenly over a number of servers. In MTS implementations, each of the servers can have access to the MTS data; however, other alternative configurations may be used instead.

[0025] The user systems **12** can be implemented as any computing device(s) or other data processing apparatus or systems usable by users to access the database system **16**. For example, any of user systems **12** can be a desktop computer, a work station, a laptop computer, a tablet computer, a handheld computing device, a mobile cellular phone (for example, a "smartphone"), or any other Wi-Fi-enabled device, wireless access protocol (WAP)-enabled device, or other computing device capable of interfacing directly or indirectly to the Internet or other network. The terms "user system" and "computing device" are used interchangeably herein with one another and with the term "computer." As described above, each user system **12** typically executes an HTTP client, for example, a web browsing (or simply "browsing") program, such as a web browser based on the WebKit platform, Microsoft's Internet Explorer browser, Apple's Safari, Google's Chrome, Opera's browser, or Mozilla's Firefox browser, or the like, allowing a user (for example, a subscriber of on-demand services provided by the system **16**) of the user system **12** to access, process and view information, pages and applications available to it from the system **16** over the network **14**.

[0026] Each user system **12** also typically includes one or more user input devices, such as a keyboard, a mouse, a trackball, a touch pad, a touch screen, a pen or stylus or the like, for interacting with a graphical user interface (GUI) provided by the browser on a display (for example, a monitor screen, liquid crystal display (LCD), light-emitting diode (LED) display, among other possibilities) of the user system **12** in conjunction with pages, forms, applications and other information provided by the system **16** or other systems or servers. For example, the user interface device can be used to access data and applications hosted by system **16**, and to perform searches on stored data, and otherwise allow a user to interact with various GUI pages that may be presented to a user. As discussed above, implementations are suitable for use with the Internet, although other networks can be used instead of or in addition to the Internet, such as an intranet, an extranet, a virtual private network (VPN), a non-TCP/IP based network, any LAN or WAN or the like.

[0027] The users of user systems **12** may differ in their respective capacities, and the capacity of a particular user system **12** can be entirely determined by permissions (permission levels) for the current user of such user system. For example, where a salesperson is using a particular user system **12** to interact with the system **16**, that user system can have the capacities allotted to the salesperson. However, while an administrator is using that user system **12** to interact with the system **16**, that user system can have the capacities allotted to that administrator. Where a hierarchical role model is used, users at one permission level can have access to applications, data, and database information accessible by a lower permission level user, but may not have access to certain applications, database information, and data accessible by a user at a higher permission level. Thus, different users generally will have different capabilities with regard to accessing and modifying application and database information, depending on the users' respective security or permission levels (also referred to as "authorizations").

[0028] According to some implementations, each user system **12** and some or all of its components are operator-configurable using applications, such as a browser, including computer code executed using a central processing unit (CPU) such as an Intel Pentium® processor or the like. Similarly, the system **16** (and additional instances of an MTS, where more than one is present) and all of its components can be operator-configurable using application (s) including computer code to run using the processor system **17**, which may be implemented to include a CPU, which may include an Intel Pentium® processor or the like, or multiple CPUs.

[0029] The system **16** includes tangible computer-readable media having non-transitory instructions stored thereon/in that are executable by or used to program a server or other computing system (or collection of such servers or computing systems) to perform some of the implementation of processes described herein. For example, computer program code **26** can implement instructions for operating and configuring the system **16** to intercommunicate and to process web pages, applications and other data and media content as described herein. In some implementations, the computer code **26** can be downloadable and stored on a hard disk, but the entire program code, or portions thereof, also can be stored in any other volatile or non-volatile memory medium or device as is well known, such as a ROM or RAM, or provided on any media capable of storing program code, such as any type of rotating media including floppy disks, optical discs, digital versatile disks (DVD), compact disks (CD), microdrives, and magneto-optical disks, and magnetic or optical cards, nanosystems (including molecular memory ICs), or any other type of computer-readable medium or device suitable for storing instructions or data. Additionally, the entire program code, or portions thereof, may be transmitted and downloaded from a software source over a transmission medium, for example, over the Internet, or from another server, as is well known, or transmitted over any other existing network connection as is well known (for example, extranet, VPN, LAN, etc.) using any communication medium and protocols (for example, TCP/IP, HTTP, HTTPS, Ethernet, etc.) as are well known. It will also be appreciated that computer code for the disclosed implementations can be realized in any programming language that can be executed on a server or other computing system such as, for example, C, C++, HTML, any other markup language, Java™, JavaScript, ActiveX, any other scripting language, such as VBScript, and many other programming languages as are well known may be used. (Java™ is a trademark of Sun Microsystems, Inc.).

[0030] FIG. **1**B shows a block diagram of example implementations of elements of FIG. **1**A and example interconnections between these elements according to some implementations. That is, FIG. **1**B also illustrates environment **10**, but FIG. **1**B, various elements of the system **16** and various interconnections between such elements are shown with more specificity according to some more specific implementations. Additionally, in FIG. **1**B, the user system **12** includes a processor system **12**A, a memory system **12**B, an input system **12**C, and an output system **12**D. The processor system **12**A can include any suitable combination of one or more processors. The memory system **12**B can include any suitable combination of one or more memory devices. The input system **12**C can include any suitable combination of input devices, such as one or more touchscreen interfaces,

keyboards, mice, trackballs, scanners, cameras, or interfaces to networks. The output system **12**D can include any suitable combination of output devices, such as one or more display devices, printers, or interfaces to networks.

[0031] In FIG. **1**B, the network interface **20** is implemented as a set of HTTP application servers $\mathbf{100}_1$-$\mathbf{100}_N$. Each application server **100**, also referred to herein as an "app server", is configured to communicate with tenant database **22** and the tenant data **23** therein, as well as system database **24** and the system data **25** therein, to serve requests received from the user systems **12**. The tenant data **23** can be divided into individual tenant storage spaces **112**, which can be physically or logically arranged or divided. Within each tenant storage space **112**, user storage **114** and application metadata **116** can similarly be allocated for each user. For example, a copy of a user's most recently used (MRU) items can be stored to user storage **114**. Similarly, a copy of MRU items for an entire organization that is a tenant can be stored to tenant storage space **112**.

[0032] The process space **28** includes system process space **102**, individual tenant process spaces **104** and a tenant management process space **110**. The application platform **18** includes an application setup mechanism **38** that supports application developers' creation and management of applications. Such applications and others can be saved as metadata into tenant database **22** by save routines **36** for execution by subscribers as one or more tenant process spaces **104** managed by tenant management process **110**, for example. Invocations to such applications can be coded using PL/SOQL **34**, which provides a programming language style interface extension to API **32**. A detailed description of some PL/SOQL language implementations is discussed in commonly assigned U.S. Pat. No. 7,730,478, titled METHOD AND SYSTEM FOR ALLOWING ACCESS TO DEVELOPED APPLICATIONS VIA A MULTI-TENANT ON-DEMAND DATABASE SERVICE, by Craig Weissman, issued on Jun. 1, 2010, and hereby incorporated by reference in its entirety and for all purposes. Invocations to applications can be detected by one or more system processes, which manage retrieving application metadata **116** for the subscriber making the invocation and executing the metadata as an application in a virtual machine.

[0033] The system **16** of FIG. **1**B also includes a user interface (UI) **30** and an application programming interface (API) **32** to system **16** resident processes to users or developers at user systems **12**. In some other implementations, the environment **10** may not have the same elements as those listed above or may have other elements instead of, or in addition to, those listed above.

[0034] Each application server **100** can be communicably coupled with tenant database **22** and system database **24**, for example, having access to tenant data **23** and system data **25**, respectively, via a different network connection. For example, one application server $\mathbf{100}_1$ can be coupled via the network **14** (for example, the Internet), another application server $\mathbf{100}_{N-1}$ can be coupled via a direct network link, and another application server $\mathbf{100}_N$ can be coupled by yet a different network connection. Transfer Control Protocol and Internet Protocol (TCP/IP) are examples of typical protocols that can be used for communicating between application servers **100** and the system **16**. However, it will be apparent to one skilled in the art that other transport protocols can be used to optimize the system **16** depending on the network interconnections used.

[0035] In some implementations, each application server **100** is configured to handle requests for any user associated with any organization that is a tenant of the system **16**. Because it can be desirable to be able to add and remove application servers **100** from the server pool at any time and for various reasons, in some implementations there is no server affinity for a user or organization to a specific application server **100**. In some such implementations, an interface system implementing a load balancing function (for example, an F5 Big-IP load balancer) is communicably coupled between the application servers **100** and the user systems **12** to distribute requests to the application servers **100**. In one implementation, the load balancer uses a least-connections algorithm to route user requests to the application servers **100**. Other examples of load balancing algorithms, such as round robin and observed-response-time, also can be used. For example, in some instances, three consecutive requests from the same user could hit three different application servers **100**, and three requests from different users could hit the same application server **100**. In this manner, by way of example, system **16** can be a multi-tenant system in which system **16** handles storage of, and access to, different objects, data and applications across disparate users and organizations.

[0036] In one example storage use case, one tenant can be a company that employs a sales force where each salesperson uses system **16** to manage aspects of their sales. A user can maintain contact data, leads data, customer follow-up data, performance data, goals and progress data, etc., all applicable to that user's personal sales process (for example, in tenant database **22**). In an example of a MTS arrangement, because all of the data and the applications to access, view, modify, report, transmit, calculate, etc., can be maintained and accessed by a user system **12** having little more than network access, the user can manage his or her sales efforts and cycles from any of many different user systems. For example, when a salesperson is visiting a customer and the customer has Internet access in their lobby, the salesperson can obtain critical updates regarding that customer while waiting for the customer to arrive in the lobby.

[0037] While each user's data can be stored separately from other users' data regardless of the employers of each user, some data can be organization-wide data shared or accessible by several users or all of the users for a given organization that is a tenant. Thus, there can be some data structures managed by system **16** that are allocated at the tenant level while other data structures can be managed at the user level. Because an MTS can support multiple tenants including possible competitors, the MTS can have security protocols that keep data, applications, and application use separate. Also, because many tenants may opt for access to an MTS rather than maintain their own system, redundancy, up-time, and backup are additional functions that can be implemented in the MTS. In addition to user-specific data and tenant specific data, the system **16** also can maintain system level data usable by multiple tenants or other data. Such system level data can include industry reports, news, postings, and the like that are sharable among tenants.

[0038] In some implementations, the user systems **12** (which also can be client systems) communicate with the application servers **100** to request and update system level and tenant-level data from the system **16**. Such requests and updates can involve sending one or more queries to tenant database **22** or system database **24**. The system **16** (for

example, an application server **100** in the system **16**) can automatically generate one or more SQL statements (for example, one or more SQL queries) designed to access the desired information. System database **24** can generate query plans to access the requested data from the database. The term "query plan" generally refers to one or more operations used to access information in a database system.

[0039] Each database can generally be viewed as a collection of objects, such as a set of logical tables, containing data fitted into predefined or customizable categories. A "table" is one representation of a data object, and may be used herein to simplify the conceptual description of objects and custom objects according to some implementations. It should be understood that "table" and "object" may be used interchangeably herein. Each table generally contains one or more data categories logically arranged as columns or fields in a viewable schema. Each row or element of a table can contain an instance of data for each category defined by the fields. For example, a CRM database can include a table that describes a customer with fields for basic contact information such as name, address, phone number, fax number, etc. Another table can describe a purchase order, including fields for information such as customer, product, sale price, date, etc. In some MTS implementations, standard entity tables can be provided for use by all tenants. For CRM database applications, such standard entities can include tables for case, account, contact, lead, and opportunity data objects, each containing pre-defined fields. As used herein, the term "entity" also may be used interchangeably with "object" and "table."

[0040] In some MTS implementations, tenants are allowed to create and store custom objects, or may be allowed to customize standard entities or objects, for example by creating custom fields for standard objects, including custom index fields. Commonly assigned U.S. Pat. No. 7,779,039, titled CUSTOM ENTITIES AND FIELDS IN A MULTI-TENANT DATABASE SYSTEM, by Weissman et al., issued on Aug. 17, 2010, and hereby incorporated by reference in its entirety and for all purposes, teaches systems and methods for creating custom objects as well as customizing standard objects in a multi-tenant database system. In some implementations, for example, all custom entity data rows are stored in a single multitenant physical table, which may contain multiple logical tables per organization. It is transparent to customers that their multiple "tables" are in fact stored in one large table or that their data may be stored in the same table as the data of other customers.

II. Instance Advisory Service

[0041] The underutilization of instances that are provided by an IaaS service in terms of their CPU/Memory/Storage usage can be problematic from a cost perspective. In particular, when an instance is purchased on a per year basis the full cost of the instance must be paid whether usage of the resources associated with the instance is maximized or not.

[0042] An approach that addresses the shortcomings of such approaches is disclosed and described herein. For example, as part of a disclosed process, in an embodiment, the underutilization of resources can be identified and a more cost-effective instance type found. More specifically, the instance type is automatically resized. In an embodiment, the resulting reduced cost to serve, can be significant if applied over a significant number of machines.

[0043] In an embodiment, IaaS space is utilized to monitor/identify/downsize instances. For example, instance usage, in particular, that related to the CPU, memory and storage is monitored. In order to identify whether the instance is eligible for resizing, metrics collected over an extended period of time are compared to a % of the maximum capacity that is available at the different dimensions (CPU/memory/storage). In an embodiment, when a service running on the instance is deemed to have acquired more infrastructure than needed, leading to higher costs, a more appropriate instance can be identified for use. In an embodiment, a rolling restart of the service (e.g., CRM service) can be performed to push the appropriate instance type to production. In an embodiment, the resultant changes in a services workload (optimizations) that reduce resource usage, can be directly reflected in cost savings.

[0044] In an embodiment, the tenant management process **110** can include an instance advisory service **200** (see FIG. **1**B). In an embodiment, the instance advisory service **200** can monitor the utilization of resources allocated to one or more instances that are part of an application cluster that provides a service, such as a CRM service to a user system **12**. In an embodiment, as part of the monitoring, the instance advisory service **200** can determine if there is an instance type with a resource profile, for at least one of the one or more instances that will result in a more optimal level of performance and cost, than does a currently used instance type. In an embodiment, the determination can be based at least in part on resource utilization.

[0045] In an embodiment, the instance advisory service **200**, can recommend that a current instance type be replaced if an instance type that is available from the IaaS can be identified that will provide a more optimal level of performance as regards a service that is provided. In an embodiment, in addition, the instance advisory service **200** can request an evaluation of the recommended instance type. In an embodiment, the evaluation can be performed as part of a canary experiment. In an embodiment, based on the results of the canary experiment, one or more instances can be replaced or not replaced with a recommended instance type.

[0046] As used herein, the performance P of an instance is intended to refer to the proximity of the utilization of its resources to a predetermined baseline or optimal level of resource utilization. In an embodiment, the resources allocated to an instance are provided at a cost C that is based at least in part on the amount of the resources that are allocated.

[0047] In an embodiment, a more optimal instance type, is an instance type that results in a level of resource utilization per resource that is closer in terms of actual resource utilization to a predetermined optimal level of resource utilization, than does the current instance type.

[0048] In an embodiment, a heuristic can be used to identify an instance type that can achieve a level of performance that falls within an acceptable range of performance levels that are representable as percentages of the baseline level of performance. In an embodiment, such percentages of the baseline level of performance can extend above and below the baseline level of performance P. In an embodiment, the baseline level of performance can have a corresponding cost C. Moreover, in an embodiment, each level of performance P within the acceptable range of performance levels can have a corresponding cost C.

[0049] In an embodiment, the baseline or optimal level of performance can be defined before a determination is made

regarding whether a more optimal instance type is available. The choice of the baseline level or optimal level of performance can be based on a variety of factors such as but not limited to a desire to accommodate peak resource utilization periods and to minimize underutilization. For example, consider an embodiment, where a baseline or optimal level of performance of 80 percent of the maximum capacity of a resource provided by an instance is selected based on such a consideration. In such an embodiment, resource utilization that falls below 80 percent of the maximum capacity can be detected and the heuristic used to determine if there is a more optimal instance type. In other embodiments, the baseline or optimal level of performance can be other percentages of the maximum capacity.

Operation

[0050] FIG. **2**A illustrates the operation of the instance advisory service **200** of FIG. **1**B. FIG. **2**A shows instance advisory service **200**, application cluster **210**, metrics service **220**, instance metadata service **230**, and instance manager **240**. In FIG. **2**A, the operations A-G illustrate the interaction of the instance advisory service **200** with other components as a part of automatically identifying and right-sizing instances.

[0051] Referring to FIG. **2**A, at A, the instance advisory service **200** queries an IaaS provider's (or other type service provider's) instance type endpoint, such as the instance metadata service **230**, in order to maintain a current list of prices per instance type. As examples, as service provider can offer various instance type products that can include features such as but not limited to term length, reservation policy, and so forth. In some examples, instances are separate stacks of hardware and software on which various resources may be operated. It should be noted that instance types can include resources that have different CPU quality (e.g., make/model) and quantity, memory, storage, and networking capabilities. In an embodiment, based on the queries, the instance advisory service **200** can maintain a comprehensive list of the pricing catalog and stock keeping unit (SKU) types.

[0052] At B, metrics related to the instance type are streamed to the metrics service **220** from the application cluster **210**. In an embodiment, the application cluster **210** includes multiple instances of containers that run an application that provides a service such as a CRM service. In an embodiment, applications typically run across one or more containers. A container is mapped to an instance type. Each resource that is used in the instance type including CPU, memory, storage, and network has a corresponding set of metrics. In an embodiment, the metrics are continuously streamed to the metrics service.

[0053] At C, the instance advisory service **200** acquires the metrics related to service that is provided by the application cluster **210** and applies a heuristic to determine based on the workload patterns of the service, characterized by CPU, memory, storage and network usage, if there is a more optimal instance type for one or more instances of the application cluster **210**. It should be appreciated that details related to an exemplary heuristic is described with reference to FIG. **2**D. In an embodiment, decisions between candidate instance types can be based on tradeoffs between P and C where, for example, among candidate instance types a first candidate may provide the highest performance at a first cost, a second candidate an intermediate performance at a second cost, and a third candidate the lowest performance at a third cost. In an embodiment, performance P and cost C percentages can have a default value, however, a service, e.g., a CRM service, etc., can override the default values based on what is important to the service. In an embodiment, the instance advisory service **200** continuously feeds the metric stream as well as the pricing catalog and performance and cost features into the heuristic, which produces a recommended instance type as an output (see FIG. **2**D).

[0054] At D, the instance advisory service **200** determines if the recommended instance type is different than one or more of the instance types that the current service, e.g., CRM service, etc., is using. In an embodiment, if the recommended instance type is different than the instance type that the current service is using, the instance advisory service **200** can request that the instance manager **240** run a canary experiment for purposes of evaluating the recommended instance type. In an embodiment, the instance manager **240** can manage any failures and rollbacks related to the canary experiment.

[0055] At E, the instance advisory service **200** sends a request to the instance manager **240** to run the canary experiment using the instance type that is recommended. In an embodiment, as part of the request, the instance advisory service **200** provides the instance manager **240** with an identifier of the service (e.g., CRM service or other service being provided) and an identifier of the instance type that is recommended. In response to the request, the instance manager **240** can provision one or more instances of the type that is recommended. In an embodiment, the provisioned instances are then directed to service the work that is fed to the application cluster **210** on a trial basis. In an embodiment, the metrics from the provisioned instances are provided to the metrics service **220**. In an embodiment, the instance advisory service **200** can create an experimental entity to track the results of the experiment.

[0056] At F, after a predetermined period of time the instance advisory service **200** checks the results of the experiment. In an embodiment, as part of checking the results of the experiment, the instance advisory service **200** queries the experiment records to identify the service and the instance type that is being evaluated. Thereafter, the instance advisory service **200** can access metrics to examine the performance and cost of the proposed instance types, and compare them to the performance and cost of the instance types currently being used. In an embodiment, if the experiment is designed to identify the best performance within a selected range of costs, and results show that an identified best performance is better than the performance of the current instance type, then the current instance type can be replaced. In addition, if the experiment is designed to optimize cost within a selected range of performances, and the identified cost is lower than that of the current instance type, then the current instance type can be replaced. In contrast, if the results show that the optimization goal is not being met, the experiment can be stopped.

[0057] At G, the instance advisory service **200** indicates to the instance manager **240** the number of instances of the application cluster **210** that should be migrated (or not migrated) to the recommended instance type, based on the results of the canary experiment. In an embodiment, the indication can be based on the results of a variety of iterations of the canary experiment. In an embodiment, the instance advisory service **200** and the instance manager **240**

can continue a canary experiment until either, the entire application cluster **210** is running on the proposed instance type, or the optimization criteria cannot be met.

[0058] FIG. **2**B shows components of the instance advisory service **200** according to an embodiment. In the FIG. **2**B embodiment, the instance advisory service **200** includes resource monitor **201**, instance type determiner **203**, and instance type recommender **205**.

[0059] Referring to FIG. **2**B, the resource monitor **201** monitors characteristics of resource utilization and cost that are associated with one or more instances that are part of a set of resources and/or part of an application cluster (e.g., application cluster **210** in FIG. **2**A). In an embodiment, the resource monitor **201** can monitor workload metrics (e.g., CPU, memory, average page time (APT), etc.), instance pricing specifications, throttling events, and desired performance and cost configurations. Additionally or alternatively, the resource monitor **201** can monitor other types of metrics and metadata.

[0060] Instance type determiner **203** determines if there is an instance type with a resource profile, for at least one of the one or more instances that will provide a more optimal level of performance and cost, based at least in part on resource utilization, than does the current instance type. In an embodiment, the instance type determiner **203** applies a heuristic to determine based on the workload patterns of a service (e.g., CRM service or other type service), characterized by CPU, memory, storage and network usage, if there is a more optimal instance type. In an embodiment, decisions between proposed instance types can be based on tradeoffs between P and C where for example among proposed instance types a first proposed instance type may provide the highest performance at a first cost, a second proposed instance type an intermediate performance at a second cost, and a third proposed instance type the lowest performance at a third cost. In an embodiment, performance P and cost C percentages can have a default value, however, services (e.g., CRM services or other type services) can override these default values based on what is important to them.

[0061] Instance type recommender **205** recommends an instance type replacement for the at least one of the one or more instances, if an instance type that can provide a more optimal level of performance and cost, is identified that is different from the instance type that the current service (e.g., CRM service or other type service that is being provided) is using. In an embodiment, the instance type recommender **205** can request a canary experiment to be executed for purposes of evaluating the recommended instance type.

[0062] FIG. **2**C shows components of the instance manager **240** according to an embodiment. In the FIG. **2**C embodiment, the instance manager **240** includes evaluation request receiver **241** and instance type provisioner **243**.

[0063] Referring to FIG. **2**C, the evaluation request receiver **241** receives (or otherwise accesses) a request to perform an evaluation of the recommended instance type from instance type recommender **205**. In an embodiment, as part of the request, the evaluation request receiver **241** receives or accesses an identifier of the service for which the instance type will be evaluated and an identifier of the instance type that will be evaluated.

[0064] The instance type provisioner **243** provisions, as part of an experiment, one or more instances of the recommended instance type to service work associated with an application cluster (e.g., **210** in FIG. **2**A). In an embodiment, the instance type provisioner **243** can provision one or more instances of the recommended type as part of a canary experiment. In an embodiment, the metrics from the instances provisioned as part of the canary experiment are provided to the metrics service (e.g., **220** in FIG. **2**A).

[0065] In an embodiment, an experiment entity can be created (e.g., by instance advisory service **200**) to track the results of the canary experiment. For example, in an embodiment, after predetermined periods of time, an experiment entity of the instance advisory service **200** can access the results of the canary experiment. In an embodiment, if the experiment is designed to optimize performance within a selected range of costs, and results show that an instance type identified as providing the best performance provides a performance better than the performance of the current instance type, then the current instance type can be replaced. In addition, if the experiment is designed to optimize cost within a selected range of performances, and the instance type identified provides a cost that is lower than that of the current instance type, then the current instance type can be replaced. However, if the results show that the optimization goal is not being met, the experiment can be stopped.

[0066] FIG. **2**D illustrates an example implementation of the instance advisory service **200** and the interaction of instance advisory service **200** with other components such as metric service **220**, instance metadata service **230**, instance manager **240**, performance cost and configuration service **250**, and service protection service **260**.

[0067] At node 1 in FIG. **2**D, resource monitor **201** receives inputs from metrics service **220**, instance metadata service **230**, performance cost configuration service **250** and service protection service **260**. As examples, the inputs to the resource monitor **201** can include but are not limited to workload metrics (e.g., CPU (e.g., processor usage or utilization, processor idle time, host system memory usage, processor time, processor time ratio, processor non-idle wait times, processor power usage or consumption, and/or the like), memory (e.g., memory utilization, memory access wait times, memory response time, storage area network (SAN) input/output (IO) measurements/metrics such as IO operations per second (IOPS)), a size of each IO request size, IO response times and/or IO latency times, IO queue sizes, and/or the like), average page time (APT), number of transactions, number or user requests, average size or user requests, average size of DB queries, number of user responses, average response time, number of requests to access individual resources or tenant data, number of incidents (failures, errors, and the like), and/or the like), instance pricing specification, throttling events, desired performance and cost configurations, and/or other metrics and/or metadata. These inputs are ingested from respective sources **220**, **230**, and **250**. At node 2 in FIG. **2**D, the inputs are provided to a feature engineering component **201***a* of the resource monitor **201**. In an embodiment, the received dataset is de-noised, correlated, and vectorized into an expected format and then persisted (e.g., stored) in a datastore **201***b* of the resource monitor **201**.

[0068] At node 3 in FIG. **2**D, the instance type determiner **203** receives feature vectors from the resource monitor **201**. In an embodiment, the feature vectors are processed by an ensemble of machine learning (ML) models of the instance type determiner **203** to understand various workload patterns. In an embodiment, reinforcement learning (RL) mod-

els **203** a ingest or otherwise receive the current workload state and reward (e.g., P/C configuration), and based on the state and/or rewards, at node 4 recommends an action that is designed to maximize the reward. In an embodiment, the reward is a desired performance (P) and cost (C). Additionally or alternatively, recurrent neural networks (RNNs) **203***b* ingest or otherwise receive the vectorized data and uses the vectorized data to understand workload patterns. The recurrent neural networks **203***b* understands or learns the workload patterns by storing the state information and performing predictions. Additionally or alternatively, baseline models **203***c* provide insights into the regular workload patterns and sets the baseline that RNNs **203***b* to consume or otherwise use to identify seasonal workload patterns. At node 4 in FIG. **2**D, the RL models **203***a* provide the recommended actions to a recommendation engine **205***a* of an instance type recommender **205**, and at node 5, the RNNs **203***b* provide the scores (e.g., identifying instance types that are more optimal than currently implemented instance types) as input to the recommendation engine **205***a* of the instance type recommender **205**. The instance type recommender **205** (or recommendation engine **205***a*) uses this information as generated by the ensemble/weighted techniques of instance type determiner **203** to recommend an instance type to the instance manager **240**. At node 6 in FIG. **2**D, the instance manager **240** provides a recommended instance type to a user, org, developer, or the like. Additionally or alternatively, the instance manager **240** triggers or otherwise causes a current instance type (e.g., a currently implemented configuration of an instance) to be replaced with a recommended instance type (e.g., a recommended instance configuration for the instance). It should be appreciated that in other embodiments, other implementations of the instance advisory service **200** can be used.

[0069] In an embodiment, the instance advisory service **200** continuously collects data related to the current state (e.g., current performance and cost metrics) and desired state (e.g., desired and/or optimal performance and cost states) that are provided to the ensemble of models of instance type determiner **203** which learn and use feedback to recommend scaling options (instance types) to the instance type recommender **205** that can improve both performance and cost.

[0070] FIG. **3**A is a flowchart of a method for automatically identifying and rightsizing instances according to an embodiment. The method includes, at **301** monitoring a resource utilization of a set of resources of one or more instances, the resource utilization corresponding to a first level of performance and cost. At **303**, based on the resource utilization, determining if there is an instance type with a resource profile, for at least one of the one or more instances that will provide a second level of performance and cost that is closer to a default or optimal level of performance and cost than the first level of performance and cost. If there is an instance type with a resource profile, for at least one of the one or more instances that will provide a second level of performance and cost, that is closer to a default or optimal level of performance and cost than the first level of performance and cost, the process proceeds to **305**. If there is not an instance type with a resource profile, for at least one of the one or more instances that will provide a second level of performance and cost, that is closer to a default or optimal level of performance and cost than the first level of performance and cost, the process proceeds to stop. At **305**, based on the determining at **303**, determining if the instance type is different from a current instance type of the at least one of the one or more instances. If the instance type is different from the current instance type of the at least one of the one or more instances the process proceeds to **307** where a recommendation is made to replace the current instance type with the more optimal instance type. If the instance type is not different from a current instance type of the at least one of the one or more instances the process proceeds to stop. At **311**, an evaluation of the recommended instance type replacement is requested.

[0071] FIG. **3**B is a flowchart of a method for performing an evaluation of a recommended instance type according to an embodiment. The method includes, at **309**, receiving a request for an instance type evaluation. At **311**, provisioning instances of the recommended type to service work associated with the application cluster.

[0072] The specific details of the specific aspects of implementations disclosed herein may be combined in any suitable manner without departing from the spirit and scope of the disclosed implementations. However, other implementations may be directed to specific implementations relating to each individual aspect, or specific combinations of these individual aspects.

[0073] Additionally, while the disclosed examples are often described herein with reference to an implementation in which an on-demand database service environment is implemented in a system having an application server providing a front end for an on-demand database service capable of supporting multiple tenants, the present implementations are not limited to multi-tenant databases or deployment on application servers. Implementations may be practiced using other database architectures, i.e., ORACLE®, DB2© by IBM and the like without departing from the scope of the implementations claimed.

[0074] It should also be understood that some of the disclosed implementations can be embodied in the form of various types of hardware, software, firmware, or combinations thereof, including in the form of control logic, and using such hardware or software in a modular or integrated manner. Other ways or methods are possible using hardware and a combination of hardware and software. Additionally, any of the software components or functions described in this application can be implemented as software code to be executed by one or more processors using any suitable computer language such as, for example, Java, C++ or Perl using, for example, existing or object-oriented techniques. The software code can be stored as a computer- or processor-executable instructions or commands on a physical non-transitory computer-readable medium. Examples of suitable media include random access memory (RAM), read only memory (ROM), magnetic media such as a hard-drive or a floppy disk, or an optical medium such as a compact disk (CD) or DVD (digital versatile disk), flash memory, and the like, or any combination of such storage or transmission devices.

[0075] Computer-readable media encoded with the software/program code may be packaged with a compatible device or provided separately from other devices (for example, via Internet download). Any such computer-readable medium may reside on or within a single computing device or an entire computer system, and may be among other computer-readable media within a system or network. A computer system, or other computing device, may include

a monitor, printer, or other suitable display for providing any of the results mentioned herein to a user.

[0076] While some implementations have been described herein, it should be understood they have been presented by way of example only, and not limitation. Thus, the breadth and scope of the present application should not be limited by any of the implementations described herein, but should be defined only in accordance with the following and later-submitted claims and their equivalents.

What is claimed is:

1. A system to provide a compute optimization service for recommending compute resource usage, the system comprising:

memory circuitry to store program code of a resource analyzer, an instance type determiner, and an instance type recommender; and

processor circuitry connected to the memory circuitry, wherein:

the processor circuitry is to operate the resource analyzer to analyze resource utilization metrics of a set of resources belonging to a set of instances, the resource utilization metrics corresponding to a first level of performance or cost;

the processor circuitry is to operate the instance type determiner to determine, based on the resource utilization metrics, a recommended instance type of the set of instances that is predicted to provide at least a second cost that is defined as more optimal by a user; and

the processor circuitry is to operate the instance type recommender to:

cause evaluation of the at least one instance having the recommended instance type when the recommended instance type is different from a current instance type of the at least one instance, and

provide, based on the evaluation, a recommendation to facilitate a replacement or resizing of the at least one instance having the current instance type with the at least one instance having the recommended instance type,

wherein the processor circuitry is configured to cause evaluation by one or more machine learning (ML) models based on the resource utilization metrics to determine workload patterns of the at least one instance,

wherein the one or more ML models further determine the recommendation based on a current workload data of the at least one instance, the determined workload patterns, and a range of desired performances and costs based on configured performance characteristics including utilization,

wherein, based on the recommendation, the at least one instance having the current instance type is replaced with or resized based on the at least one instance having the recommended instance type, and

wherein one or more workloads of the at least one instance are executed using the recommended instance type.

2. The system of claim **1**, wherein the evaluation includes an evaluation of a performance or cost of the recommended instance type using workload data of the at least one instance.

3. The system of claim **1**, wherein the recommendation includes one or more recommendations to reduce a cost of the current instance type or one or more recommendations to improve a performance of one or more workloads of the at least one instance.

4. The system of claim **1**, wherein the processor circuitry is to operate the instance type determiner to operate the one or more ML models to determine the recommended instance type.

5. The system of claim **4**, wherein the evaluation includes a comparison of the first level of performance or cost with the second level of performance or cost, and a decision to replace the at least one instance having the recommended instance type is made based on the comparing.

6. The system of claim **1**, wherein the second level of performance or cost that is more optimal than the first level of performance or cost is identified from a specified range of performances or a specified range of costs.

7. The system of claim **1**, wherein the second level of performance or cost that is more optimal than the first level of performance or cost is identified from a specified performance and a specified range of costs.

8. The system of claim **1**, wherein the recommended resource type includes a resource profile, and the resource profile includes a specified amount of resources to be allocated to each resource with the recommended instance type.

9. A non-transitory computer-readable memory (NT-CRM) comprising instructions stored thereon that, in response to execution by a processor, are operable to cause the processor to:

analyze resource utilization metrics of a set of resources belonging to a set of instances, wherein the resource utilization metrics correspond to a first level of performance or cost;

determine, based on the analysis, a recommended instance type of the set of instances that is predicted to provide at least a second cost that is defined as more optimal by a user;

cause evaluation of the at least one instance with the recommended instance type when the recommended instance type is different from a current instance type of the at least one instance;

provide, based on the determination, a recommendation to facilitate a replacement or resizing of the current instance type of the at least one instances with the recommended instance type for the at least one instance;

based on the resource utilization metrics, cause evaluation by one or more machine learning (ML) models to determine workload patterns of the at least one instance;

determine, with the one or more ML models, the recommendation based on a current workload data of the at least one instance, the determined workload patterns, and a range of desired performances and costs based on configured performance characteristics including utilization;

replace, based on the recommendation, the at least one instance having the current instance type with or resized based on the at least one instance having the recommended instance type; and

execute one or more workloads of the at least one instance using the recommended instance type.

10. The NTCRM of claim 9, wherein the determination of the recommended resource type is based on outputs of the one or more ML models.

11. The NTCRM of claim 9, wherein:

the evaluation includes an evaluation of a performance or cost of the recommended instance type using workload data of the at least one instance; and

the recommendation includes one or more recommendations to reduce a cost of the current instance type or one or more recommendations to improve a performance of one or more workloads of the at least one instance.

12. The NTCRM of claim 9, wherein execution of the instructions is to cause the processor to compare the first level of performance or cost with the second level of performance or cost.

13. The NTCRM of claim 12, wherein a decision to replace the one or more instances is made based on the comparing.

14. The NTCRM of claim 9, wherein the second level of performance or cost that is more optimal than the first level of performance or cost is identified from a specified range of performances or a specified range of costs.

15. The NTCRM of claim 9, wherein the second level of performance or cost that more optimal than the first level of performance or cost is identified from a specified performances and a specified range of costs.

16. The NTCRM of claim 9, wherein the recommended resource type includes a resource profile, and the resource profile includes a specified amount of resources to be allocated for each resource having the recommended resource type.

17. A computer-implemented method for providing a compute optimization service for recommending compute resource usage, the method comprising:

analyzing resource utilization metrics of a set of resources belonging to a set of instances, wherein the resource utilization metrics correspond to a first level of performance or cost;

determining, based on the analyzing, a recommended instance type of the set of instances that is predicted to provide at least a second cost that is defined as more optimal by a user;

causing evaluation of the at least one instance having the recommended instance type when the recommended instance type is different from a current instance type of the at least one instance;

provide, based on the determining, a recommendation to facilitate a replacement or resizing of the at least one instance having the current instance type with the at least one instance having the recommended instance type;

based on the resource utilization metrics, cause evaluation by one or more machine learning (ML) models to determine workload patterns of the at least one instance;

determine, with the one or more ML models, the recommendation based on a current workload data of the at least one instance, the determined workload patterns, and a range of desired performances and costs based on configured performance characteristics including utilization;

replace or resize, based on the recommendation, the at least one instance having the current instance type based on the at least one instance having the recommended instance type; and

execute one or more workloads of the at least one instance using the recommended instance type.

18. The method of claim 17, further comprising:

operating the one or more ML models to determine the recommended instance type.

19. The method of claim 17, wherein:

the evaluation includes an evaluation of a performance or cost of the recommended instance type using workload data of the at least one instance; and

the recommendation includes one or more recommendations to reduce a cost of the current instance type or one or more recommendations to improve a performance of one or more workloads of the at least one instance.

20. The method of claim 17, further comprising:

comparing the first level of performance and cost with the second level of performance or cost.

21. The method of claim 20, wherein a decision to replace the one or more instances is made based on the comparing.

22. The method of claim 17, wherein the second level of performance or cost that is more optimal than the first level of performance or cost is identified from a specified range of performances and the specified cost.

23. The method of claim 17, wherein the second level of performance or cost that is more optimal than the first level of performance or cost is identified from a specified performance and the specified range of costs.

24. The method of claim 17, wherein the recommended resource type includes a resource profile, and the resource profile includes a specified amount of resources that should be allocated for each resource of an instance having the recommended resource type.

* * * * *