US 20120042079A1

(19) **United States**

(12) **Patent Application Publication**

Ackerman et al.

(10) **Pub. No.: US 2012/0042079 A1**

(43) **Pub. Date: Feb. 16, 2012**

(54) **TECHNIQUES FOR PROVIDING SERVICES AND ESTABLISHING PROCESSING ENVIRONMENTS**

(76) Inventors: **Mark D. Ackerman**, Eagle Mountain, UT (US); **Stephen R. Carter**, Spanish Fork, UT (US)

(21) Appl. No.: **13/278,443**

(22) Filed: **Oct. 21, 2011**

Related U.S. Application Data

(63) Continuation of application No. 10/880,224, filed on Jun. 29, 2004, now Pat. No. 8,069,443.

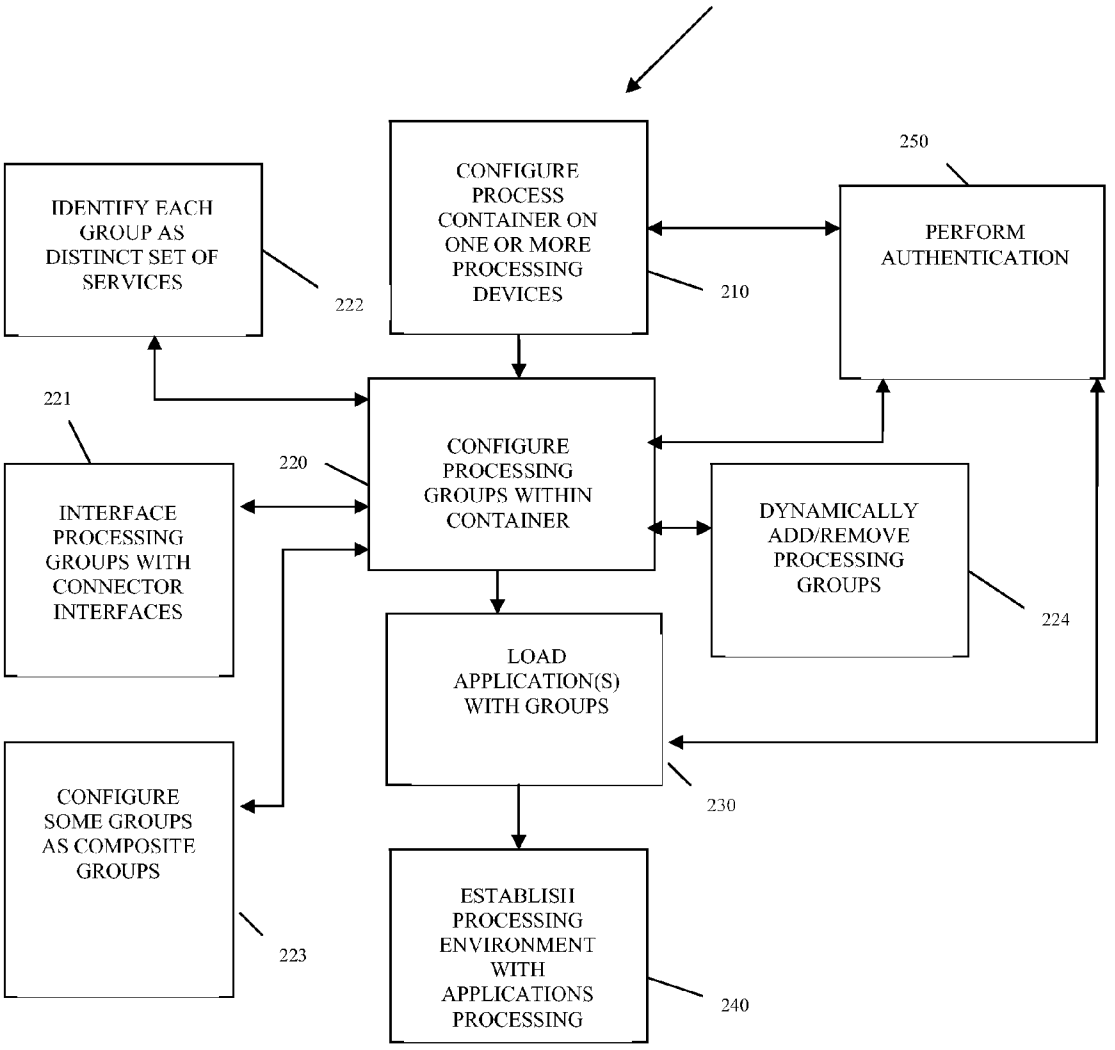
Publication Classification

(51) **Int. Cl.**
G06F 15/16 (2006.01)

(52) **U.S. Cl.** **709/226**

(57) **ABSTRACT**

Techniques are provided for the delivery of client services and for the establishment of client processing environments. A client receives services within a processing environment which is defined by a processing container. The processing container includes one or more processing groups, and each processing group has a particular context that supports one or more applications or services which are processing within that context. The processing groups communicate with one another via connector interfaces included within the processing container. Services and processing containers can be dynamically added or removed from the processing container.



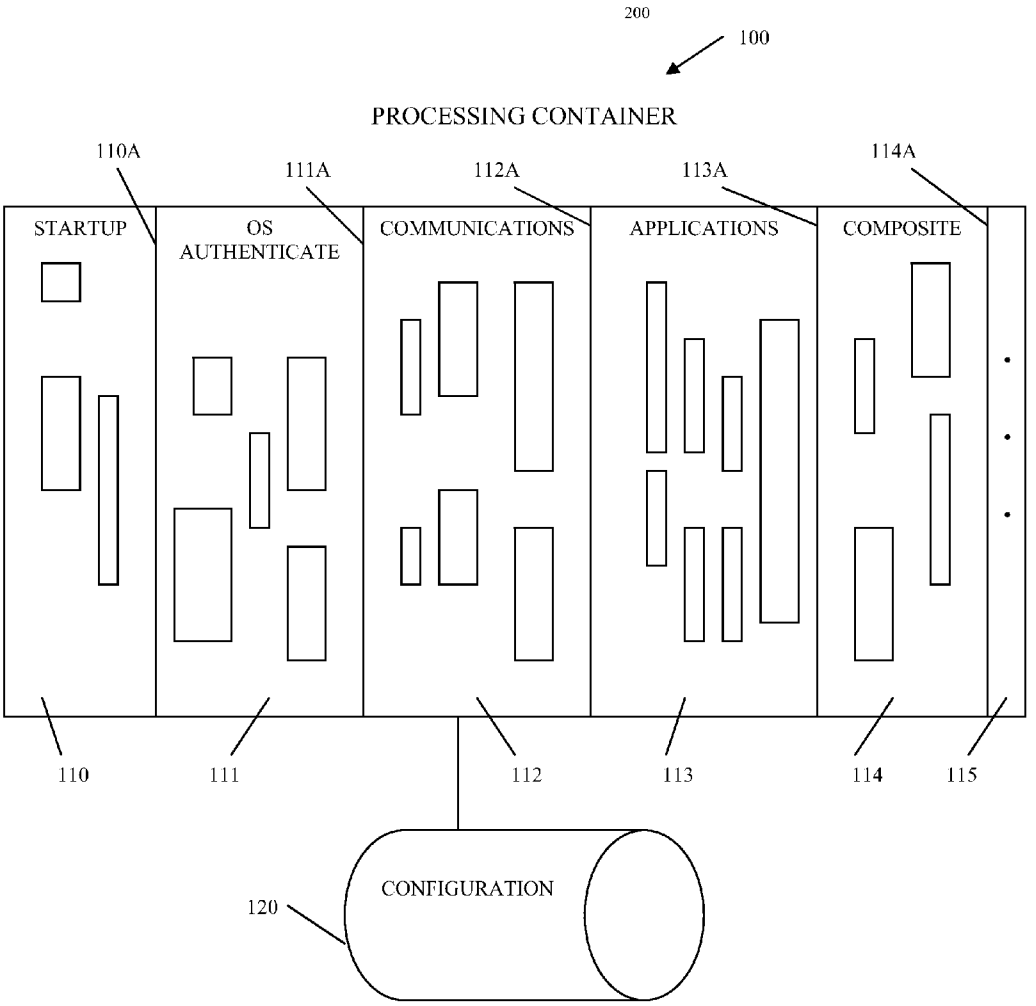

200
100
PROCESSING CONTAINER
110A
111A
112A
113A
114A
STARTUP
OS AUTHENTICATE
COMMUNICATIONS
APPLICATIONS
COMPOSITE
110
111
112
113
114
115
CONFIGURATION
120

FIG. 1

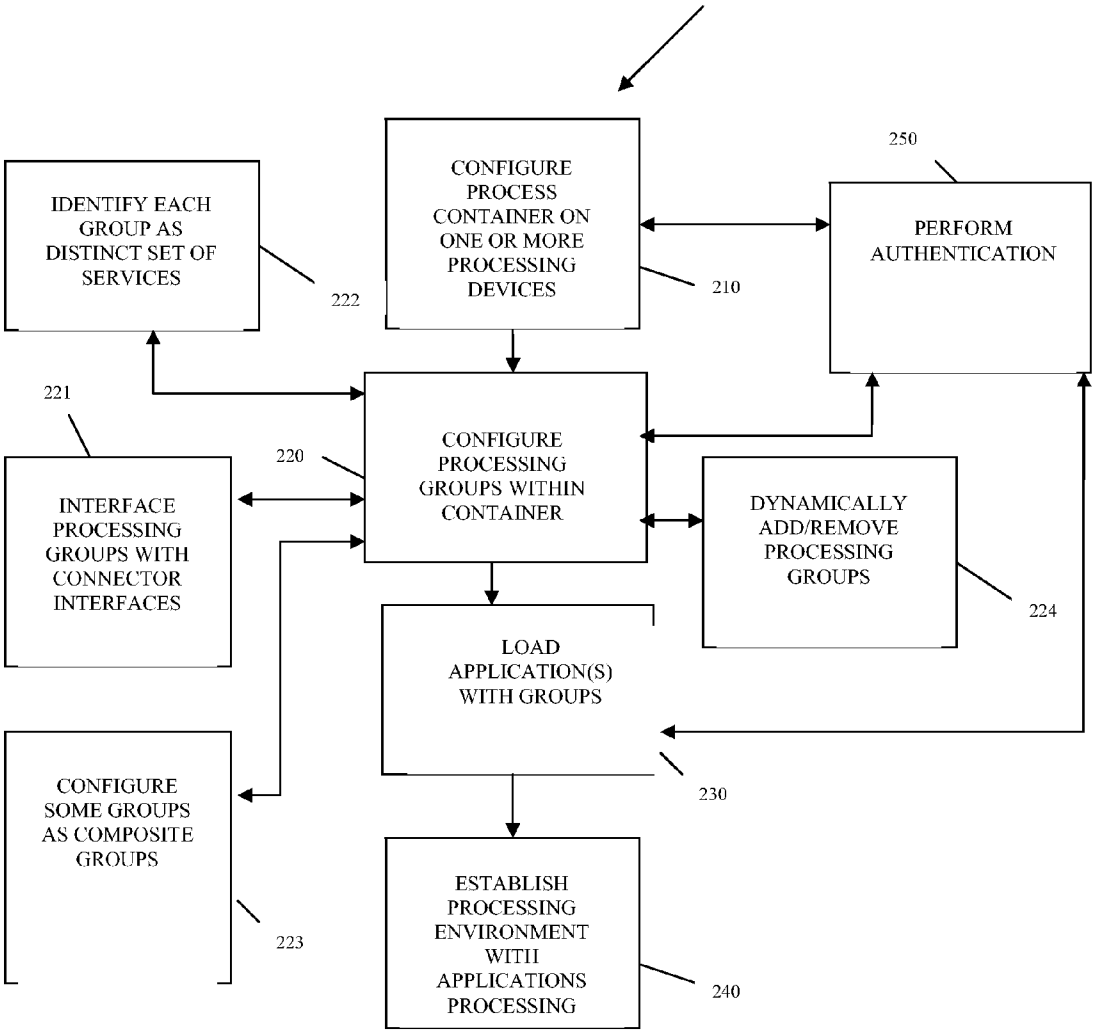

CONFIGURE PROCESS CONTAINER ON ONE OR MORE PROCESSING DEVICES
210
250
PERFORM AUTHENTICATION
IDENTIFY EACH GROUP AS DISTINCT SET OF SERVICES
222
221
CONFIGURE PROCESSING GROUPS WITHIN CONTAINER
220
INTERFACE PROCESSING GROUPS WITH CONNECTOR INTERFACES
DYNAMICALLY ADD/REMOVE PROCESSING GROUPS
224
LOAD APPLICATION(S) WITH GROUPS
230
CONFIGURE SOME GROUPS AS COMPOSITE GROUPS
223
ESTABLISH PROCESSING ENVIRONMENT WITII APPLICATIONS PROCESSING
240

FIG. 2

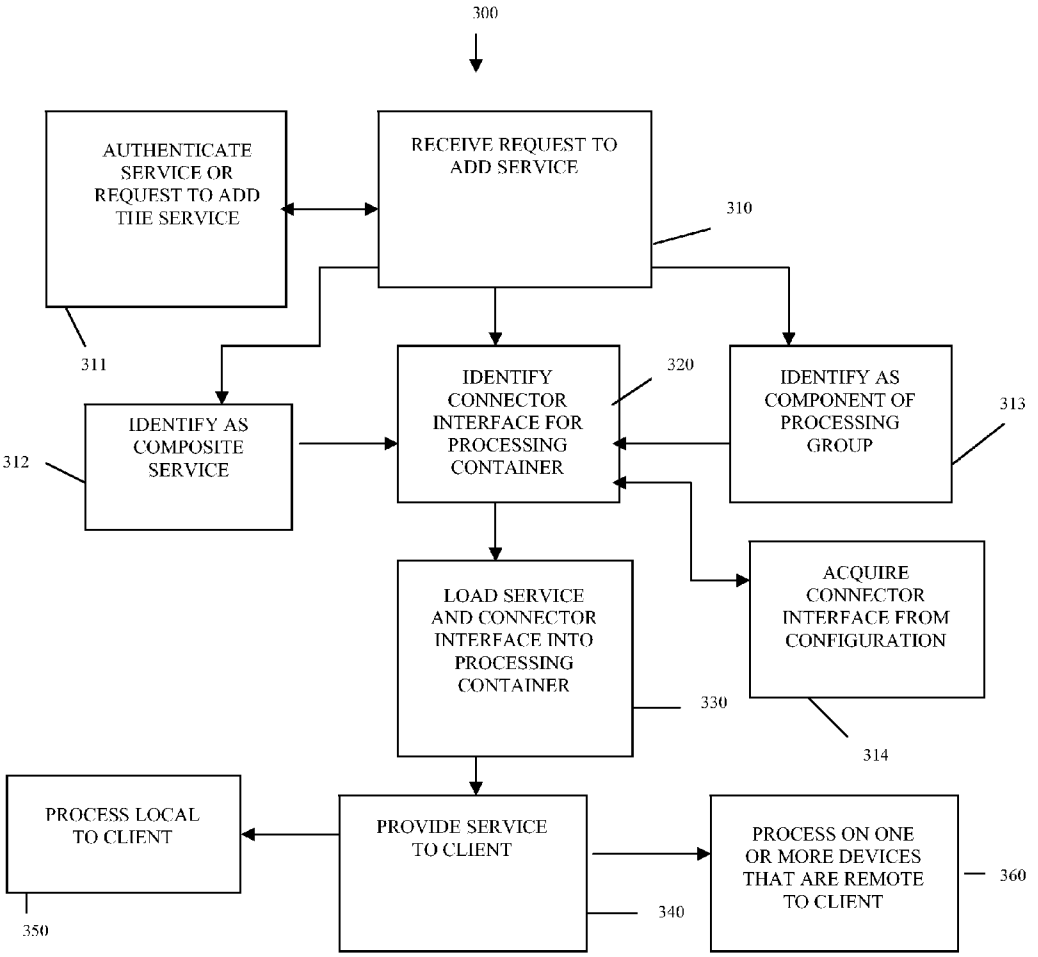

300
AUTHENTICATE SERVICE OR REQUEST TO ADD THE SERVICE
RECEIVE REQUEST TO ADD SERVICE
310
311
IDENTIFY AS COMPOSITE SERVICE
312
IDENTIFY CONNECTOR INTERFACE FOR PROCESSING CONTAINER
320
IDENTIFY AS COMPONENT OF PROCESSING GROUP
313
LOAD SERVICE AND CONNECTOR INTERFACE INTO PROCESSING CONTAINER
330
ACQUIRE CONNECTOR INTERFACE FROM CONFIGURATION
314
PROCESS LOCAL TO CLIENT
350
PROVIDE SERVICE TO CLIENT
340
PROCESS ON ONE OR MORE DEVICES THAT ARE REMOTE TO CLIENT
360

FIG. 3

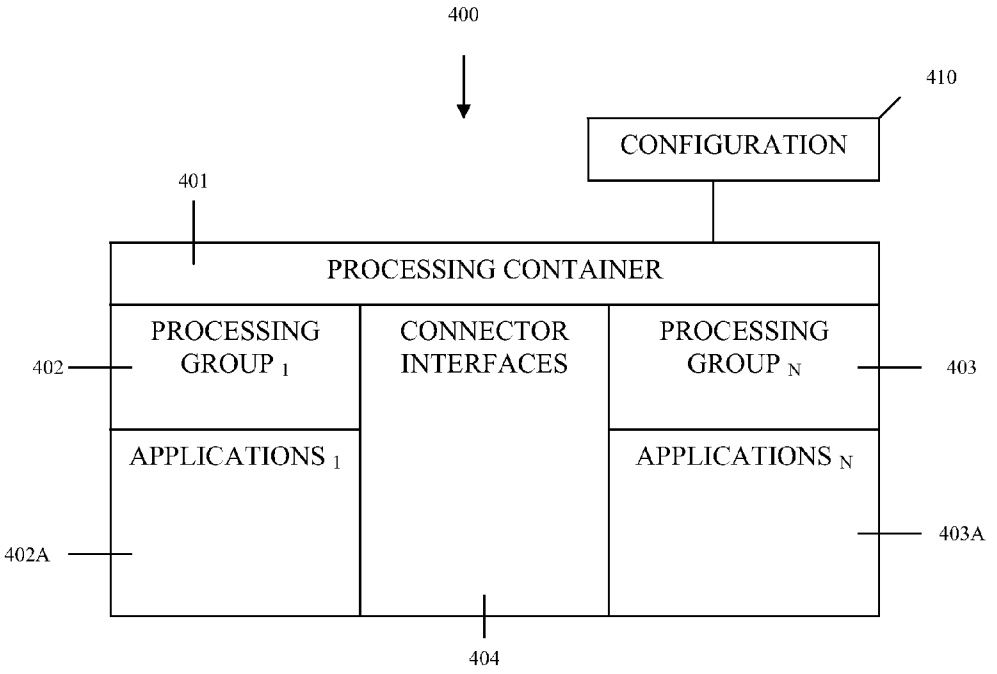

400
410
CONFIGURATION
401
PROCESSING CONTAINER
PROCESSING GROUP 1
CONNECTOR INTERFACES
PROCESSING GROUP N
402
403
APPLICATIONS 1
APPLICATIONS N
403A
402A
404

FIG. 4

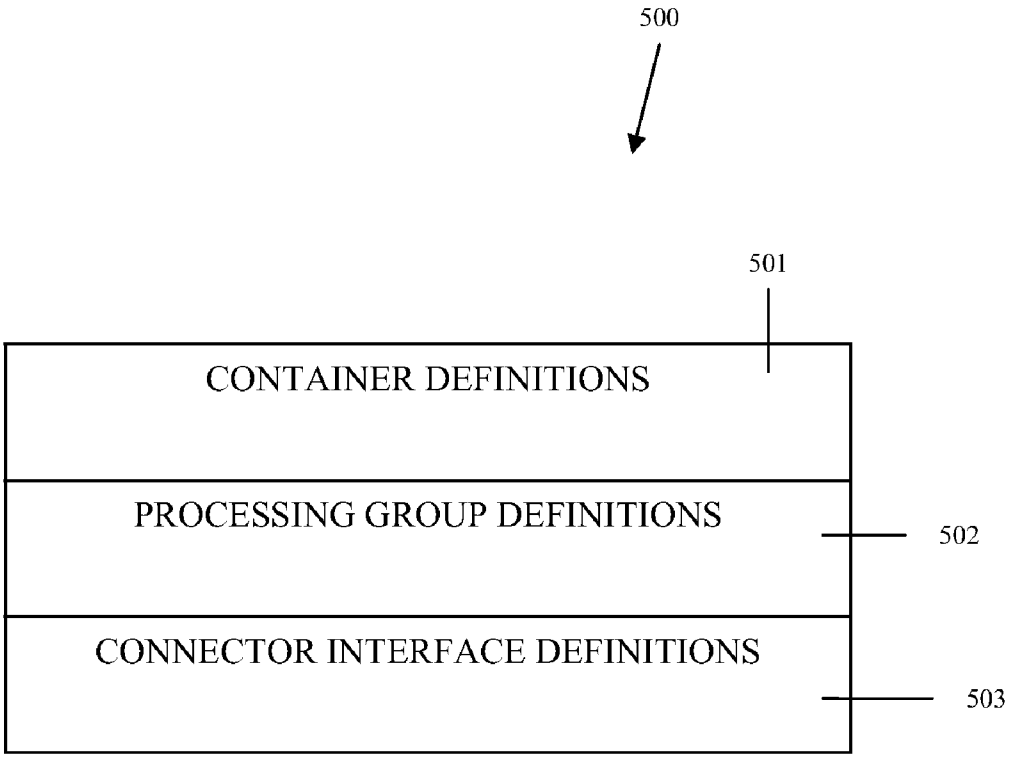

500
501
CONTAINER DEFINITIONS
PROCESSING GROUP DEFINITIONS
502
CONNECTOR INTERFACE DEFINITIONS
503

FIG. 5

TECHNIQUES FOR PROVIDING SERVICES AND ESTABLISHING PROCESSING ENVIRONMENTS

RELATED APPLICATIONS

[0001] The present application is co-pending with, claims priority to, and is a continuation of U.S. patent application Ser. No. 10/880,224, entitled: "Techniques for Providing Services and Establishing Processing Environments," filed on Jun. 29, 2004, which presently stands allowed; and the disclosure of which is incorporated by reference herein in its entirety.

FIELD OF THE INVENTION

[0002] The invention relates generally to networking and more specifically to techniques for providing client or server services and establishing processing environments for clients or servers over a network.

BACKGROUND OF THE INVENTION

[0003] Attempts have been made to integrate software over the Internet, such that services need not physically reside or be installed on a client in order for a client to acquire, process, and benefit from those services. In some cases, the client dynamically acquires the services and locally processes them. This technology is nascent and is gradually beginning to gain ground in the industry. The basic idea is that services are registered, located, acquired, and processed on demand. Services may be processed local to the client or remote from the client.

[0004] The idea of creating an Integrated Development Environment (IDE) has been around for some time and recent developments have held some promise that at these technologies will catch on and eventually become pervasive. Largely, efforts to integrate services has been slow, since in most instances each service needs to abstract its own interfaces and make them available for use by a variety of other services. Additionally, the services themselves must be available to process on a variety of operating system (OS) platforms. Consequently, making legacy services available in an IDE has been and continues to be problematic. Thus, services are still largely provided via dedicated servers or as appliances. That is, developers continue to view services as either an appliance application or as a server application.

[0005] One problem with the newer and older IDE models for providing services is that any particular client is difficult to manage and support, because the client's services which may originate from a variety of disparate platforms. In fact, the services may not even be within the management control of an administrator for a particular client. Additionally, some client services may not be compatible or interface with other client services.

[0006] Accordingly, management and integration has been more easily achieved with physical hardware, such as servers that are integrated in a rack with one another. However, physical devices require a definite physical location, have moving parts that can break, and cannot be easily transported from place to place as needed. The benefits of having services within a client's physical environment are that the services can be centrally managed, distributed, and controlled by administrators. Another significant benefit is that security can be better enforced in a centrally distributed physically controlled processing environment.

[0007] In fact, a more optimal solution to an IDE is a technique that is capable of centrally managing client services as if they were being locally stored and administered while at the same time having no requirement that those services be associated with traditional servers or appliances.

[0008] Correspondingly, techniques are described herein for providing services and establishing processing environments over a network, where these techniques are capable of being centrally controlled, managed, and administered.

SUMMARY OF THE INVENTION

[0009] In various embodiments of the invention, techniques are presented for establishing client processing environments and for delivering services to the client. A processing container defines a plurality of unique processing contexts, which represent processing groups. Applications process within the contexts. Communication between different contexts of the processing container is achieved via one or more connector interfaces.

[0010] More specifically, and in one embodiment, a method for establishing a processing environment is presented. A processing container is configured on one or more processing devices. One or more processing groups are configured within the processing container, where each processing group has a unique context. One or more applications are loaded within each processing group's context and a processing environment established. The processing environment has the one or more applications processing within their respective processing group's context and within the container.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] FIG. **1** is diagram of a processing architecture, according to an example embodiment of the invention;

[0012] FIG. **2** is a flowchart representing a method for establishing a processing environment, according to an example embodiment of the invention;

[0013] FIG. **3** is a flowchart representing a method for providing a service to a client, according to an example embodiment of the invention;

[0014] FIG. **4** is a diagram representing a processing system, according to an example embodiment of the invention.

[0015] FIG. **5** is a diagram representing configuration information that defines a processing environment, according to an example embodiment of the invention.

DETAILED DESCRIPTION OF THE INVENTION

[0016] In various embodiments of the invention, the phrase "processing container" is used. A processing container is a logical environment, such as a virtual machine, logical network, or a logical rack ("soft rack") that holds and processes a variety of logical devices. These devices are actually processing groups, such that each processing group has a unique processing context which supports processing specific applications requiring the unique context. The term application is synonymous herein and below with the term service. The processing groups are similar to server blades that are designed to perform specific services and are installed in racks having a plurality of other blades. In this sense, the processing groups can be viewed as "soft blades."

[0017] Thus, with embodiments of this invention, the processing container and its processing groups are logical or virtual associations that are not tied to any particular physical

location or hardware device. Accordingly, a processing container and its processing groups are hardware and location independent.

[0018] Each processing group includes one or more connector interfaces, the connector interfaces permit one processing group to interface and communicate with another different processing group of the processing container. Connector interfaces are aware of the applications processing within their respective processing groups and understand what types of output the applications produce and what types of input the applications require for processing. In this way, the connector interfaces can interface between applications having context in one processing group of the processing container with different applications having different contexts in different processing groups. It is noted, that in the present described arrangement the applications themselves need not be aware of the processing container, the connector interfaces, and the processing groups. That is, legacy and existing applications can be installed as soft blades and processed within particular processing groups of a processing container.

[0019] In one embodiment, the techniques presented herein are incorporated into network arrangements and products. These techniques create processing environments for clients of a network and define how services are delivered, processed, and interact with one another for any particular client. The techniques provide for centralized management and integration of distributed services.

[0020] It should be noted that a client can also be a server in any particular context. That is, a single client can serve as a server device for a plurality of other clients or servers. Thus, the use of the term client is not restricted to an end-user's processing device, and the term client is synonymous with server as used herein and below.

[0021] FIG. **1** is an example processing architecture **100** for providing services and establishing processing environments, according to an example embodiment of the invention. The architecture **100** is presented for purposes of illustration only, since one of ordinary skill in the art readily appreciates that other configurations can be achieved without departing from the teachings that are presented herein and below. The processing architecture **100** (herein after "processing container") provides a logical processing environment for one or more clients of a network.

[0022] The processing container **100** includes a plurality of processing groups **110-115**; each processing group having one or more connector interfaces **110A-114A** (may be depicted as a group of plugs, in a certain other configurations). The processing container **100**, the processing groups **110-115**, and the connector interfaces **110A-114A** are defined via a configuration **120**. The configuration **120** can be represented in formal languages, such as Extensible Markup Language (XML), and other configuration data.

[0023] The processing container **100** is broken up into a series of contextually similar regions. Each of these regions represents one of the distinct processing groups **110-115**. Within each processing group **110-115** there are a number of rectangular boxes, these boxes are intended to graphically represent applications that are available within that particular processing group **110-115**. A particular application has context and can process within its assigned processing group **110-115**.

[0024] As an example, consider that the "startup" processing group **110** includes a number of applications related to Power-On Self-Test (POST) diagnostics. These applications perform startup diagnostics to ensure that the processing container **100** is initialized and processing properly when a client starts up. Other applications within the startup processing group **110** might include Kernel operations for determining load, memory usage, etc. The OS Authenticate processing group **111** may include a myriad of authentication applications, such as on-box authentication, off-box authentication, trust authentication, public-private key pair authentication, password authentication, biometric authentication, and the like.

[0025] Continuing with the present example, the Communications processing group **112** may include applications for layer 0 communications, layer 1 communications, layer 2 communications, etc. The Applications processing group **113** may include word processing applications, spreadsheet applications, statistical applications, and the like. The Composite processing group **114** includes applications or systems that are more complex and may interact with various other processing groups **110-115**. A Composite processing group **114** may include a Virtual Private Network (VPN) application, a Content Distribution Network (CDN) application, proxy applications (forward, reverse, transparent, etc.) and the like. A Composite group **114** application interacts with one or all the other processing groups **110-115**. However, it should be noted that each of the processing groups **110-115** are capable of communicating with other processing groups **110-115**; thus, a composite group **114** is not the only processing group **110-114** with this capability. There is no limit to the number and types of processing groups **110-115** which may populate a processing container **100**, thus other processing groups **115** can be established within the processing container **100**.

[0026] Each processing group **110-115** can interact with other processing groups **110-115** via connector interfaces **110A-114A**. The connector interfaces **110A-114A** are defined in the configuration **120** and may be expressed in XML, Simple Object Access Protocol (SOAP), Remote Procedure Calls (RPC), etc. In other embodiments, the connector interfaces **110A-1145A** can be expressed as configuration or initialization files (e.g., ".conf" or ".ini"). A connector interface defines common events, parameters, and output of the processing groups **110-115**. This can be used by the processing groups **110-115** to interface with one another and convert or translate data from one application in one processing group **110-115** to different data consumed by a different application in a different processing group **110-115**. The connector interfaces **110A-114A** provide for interaction of an operation for the processing container **100**.

[0027] In one embodiment, the connector interfaces **110A-114A** are Application Programming Interfaces (APIs) or protocols associated with the individual applications, which permit the input and output of the applications to be communicated generically within the processing groups **110-115**. The configuration information **120** then permits the generically expressed data to be communicated to other connector interfaces **110A-114A** for purposes of performing some operation within the processing container **100**.

[0028] The processing groups **110-115** are designed to provide a service or a combination of services. Thus, these processing groups **110-115** may perform a function, provide a function, or in some cases provide resource specifications. For example, a processing group **110-115** may add a user limit to a mail system, such that a previous soft limit for total number of users on an electronic mail system is dynamically

increased. As another example, a licensing processing group **110-115** may evaluate and enforce licensing restrictions for a variety of other processing groups **110-115**.

[0029] The processing container **100** does not need to reside on a single processing device (although in some embodiments it can). That is, the processing container **100** is logical and is driven by the configuration **120** and can be distributed across a variety of devices over a network. Any particular client's processing environment is defined by its configuration **120** and corresponding processing container **100**. Clients dynamically add and remove services (processing groups **110-115**) by adjusting their configurations. Services are integrated and distributed via the processing container **100**. This permits centralized management and distribution of services in a logical fashion in much the same way services have previously been administered in a physical fashion, although without the previous limitations associated with physical configurations and physical devices (e.g., servers and appliances).

[0030] FIG. **2** is a flowchart of a method **200** for establishing a processing environment for a client over a network. The method **200** (hereinafter "processing") is implemented in a machine-readable and accessible medium. In one embodiment, the processing is a driver application that processes on the client or on a server on behalf of the client. The driver application consumes configuration data and operations of the client in order to process services on behalf of the client. The actual physical location of the services can vary and in many instances is remote from the client and the driver application. In one embodiment, the processing utilizes the architecture **100** of FIG. **1**.

[0031] Initially, at **210**, a processing container is configured for a client on one or more processing devices. In one embodiment, the processing container is wholly contained on a client appliance device. In an alternative embodiment, the processing container is distributed across a variety of processing devices over a network; one of these devices may also include the client appliance device. In one embodiment, the processing container is configured over devices that are distributed over multiple networks, such as a Local Area Network (LAN) and a Wide Area Network (WAN, e.g., Internet). In one embodiment, the processing container is configured based on configuration data provided for defining the processing container.

[0032] At **220**, processing groups within the processing container are configured. Each processing group has a particular processing context and is designed to provide a particular service or suite of similar services to the client. At **221**, the processing groups are interfaced to and include one or more connector interfaces. The connector interfaces are APIs or protocols that permit interaction with applications included within each of the processing groups. In some cases, these APIs or protocols abstract and normalize communications occurring within any particular processing group into a generic set of data that can be processed and recognized by other APIs and protocols associated with other connector interfaces of the processing container.

[0033] At **222**, in some embodiments, each processing group is identified during configuration as a distinct set of services. Moreover, at **223**, some of the processing groups can be identified as composite processing groups or services. This means that the composite processing groups include a variety of services and may interact with a variety of other processing groups and their contexts. Example composite processing groups can include VPNs, CDNs, forward proxies, transparent proxies, reverse proxies, and the like.

[0034] At **224**, during operation of the processing container where applications are processed on behalf of a client, a number of the processing groups can be dynamically added to or removed from the processing container. In this manner, a client's processing environment can be dynamically administered on its behalf without having to shut down the client and restart it, which has conventionally been the case.

[0035] At **230**, a variety of applications are loaded into each of the processing group's contexts. Each application loaded within a particular processing group is associated with a type of service (e.g., communication, application, composite). Thus, each application has context to process within in a certain processing group. Assignment of an application to a processing group can be achieved via configuration data that identifies the application and its processing group within the processing container.

[0036] Once the processing container and the processing groups are configured and the applications loaded into their proper processing group contexts; a processing environment is fully established for a client, at **240**. The client is now free to interact with its services (processing groups) and interaction across processing groups is achieved via the connector interfaces.

[0037] The processing environment established by the processing container presents a number of novel benefits. For example, a client may purchase a service, such as a composite processing group representing a transparent proxy where initially the client only desires minimal features of the proxy, such as access control to external web sites. As the client's needs increase, other features of that transparent proxy can be activated or augmented with other services in a dynamic fashion. For example, the transparent proxy may add VPN and CDN services. Of course a variety of other examples are possible with the processing container and all such variations are intended to fall within the generous scope of this invention.

[0038] Moreover, during any particular processing point associated with method **200** of FIG. **2**, authentication can be enforced at **250**. Thus, the initial client processing container can use authentication, the processing groups can use authentication, and the loaded applications can use authentication. This permits security to be flexibly enforced, such that even though the processing container is logical and distributed it can be authenticated and each of its components can be authenticated based on the desires of a network administrator. In fact, the authentication technique can itself be a processing group and a service to the client within the processing container.

[0039] FIG. **3** is a flowchart of a method **300** for providing a service to a client over a network. The method **300** (herein after "processing") is implemented in a machine-readable and accessible medium and is accessible over a network. In one embodiment, the method **300** utilizes the processing of method **200** of FIG. **2** and the underlying architecture **100** of FIG. **1**. The processing represents one aspect of how a client dynamically requests a new service within its processing environment via a processing container.

[0040] Initially, a client is processing within an environment that was established in manners described above with respect to method **200** and FIG. **2**. That is, the client's processing environment is defined and driven by a processing container. At some point during the client's processing, the

client requests, at **310**, that a new service be added to its processing container. In some embodiments, at **311**, this new service or the request itself can be authenticated by another service included within the client's existing processing container. The request is not honored and the new service not added to the client's processing container if they are not properly authenticated. In another embodiment, the processing container's configuration data may require that client requests or new services be authenticated, and the processing container may actually perform the authentication used.

[0041] In some embodiments, at **312**, the requested service is identified as a composite service (e.g., VPN, CDN, forward proxies, transparent proxy, reverse proxy, etc.). In other embodiments, at **313**, the requested service is identified as a component of a processing group, where that processing group is itself a service. In other words, the requested service may be a component (new feature) of an existing service.

[0042] At **314**, once the service is identified a connector interface is acquired for that service. In some cases, this connector interface is defined in structured manner and represented in XML, SOAP, RPC, ini files, conf files, etc. The connector interface is an API or protocol that permits interaction with the service. The API also abstracts the interaction into a generic format that can be communicated within the processing container between services (processing groups). In this manner, the requested service, which is being added, is integrated within the processing container. Any cross interaction between disparate services represents an operation or function of the processing container.

[0043] Once the proper connector interface and new service are identified, the service and its connector interface are loaded into the processing container at **330**. At this point, the new service is ready for use by the client. The new service was dynamically installed by the processing and the client did not require a shut down before that new service was fully available for use by the client.

[0044] In one embodiment, the initial request received for the service by the client may in fact be a generic query for a generic service. The processing can interact with other services to execute that query and locate a suitable service which satisfies the client's request. For example, the client may request a service for performing statistical processing; in response to this request, the processing locates a suitable service, such as SAP, based on the configuration of the client's processing container. The SAP is then acquired (perhaps purchased using another processing container service) and loaded into the client's processing container.

[0045] Once the requested service is loaded into the client's processing container, the service is provided to the client at **340**. In some cases, this may mean that the service is processed locally on an appliance of the client at **350**. In other cases, this may mean that the service is processed on one or more devices that are remote and external to the client at **360**. The remote processing of the requested service can occur on devices within the client's LAN or can occur on devices that are external or remote to the client which exist on a WAN (e.g., Internet).

[0046] In some embodiments, a client's request for a service may be associated with a service that already exists within the client's processing container but has not yet been activated. This may be a configuration established by a vendor of a service or by an administrator of the client. It may be done for purposes of efficiency or for purposes of easy installation and access to the service when dynamically desired by the client. In these embodiments, the service does not have to be acquired before it is loaded; it is just dynamically activated with a key, such as a licensing key and the like.

[0047] FIG. **4** is a diagram of a processing system **400**, according to an example embodiment of the invention. The processing system **400** is implemented in a machine-accessible and readable medium and is processed over a network. In one embodiment, the processing system **400** is initially established and interacted with in the manners described above with respect to methods **200** and **300** of FIGS. **2** and **3**, respectively.

[0048] It is noted that FIG. **4** is presented for purposes of illustration and comprehension and that the processing system **400** does not have to be contiguously contained within any single processing device. That is, the processing system **400**, in some embodiments, is distributed across a plurality of processing devices and logically associated and controlled by methods similar to methods **200** and **300** of FIGS. **2** and **3**, respectively.

[0049] The processing system **400** includes a processing container **401**, one or more processing groups **402**-**403**, and one or more connector interfaces **404**. Each processing group **402** or **403** includes applications **402**A and **403**A, respectively. The applications **402**A and **403**A combine within their respective processing groups **402** and **403** to provide a service or set of similar services to a client within each unique processing group **402** or **403**.

[0050] The processing groups **402** and **403** represent contexts within the processing container **401**. Certain applications **402**A and **403**A are operable to be processed within their contexts but not outside their contexts. Each processing group **402** and **403** can communicate with its applications **402**A and **403**A and with other processing groups **402** and **403** via the connector interfaces **404**.

[0051] The connector interfaces **404** are APIs or protocols which permit interaction with the applications **402**A and **403**A via the processing groups **402** and **403**. By interaction it is meant that input and output data associated with the applications **402**A and **403**A can be interpreted and provided via the connector interfaces **404**. The input and output data can be normalized or made generic and communicated to other connector interfaces **404**, where it can be communicated to other applications **402**A and **403**A. Thus, the connector interfaces **404** serve as bridges for integrating the applications **402**A and **403**A. It should be noted, that the applications **402**A and **403**A themselves may not need to be modified to be integrated into the processing container **401**; rather, a specialized connector interface **404** can be provided to integrate a particular application **402**A and **403**A, if desired.

[0052] In some embodiments, the processing system **400** includes configuration data **410**. The configuration data defines the processing container **401**, the processing groups **402** and **403**, the applications **402**A and **403**A, and the connector interfaces **404**. In order, to remove or add processing groups **402** and **403**, applications **402**A and **403**A, and connector interfaces **404** the configuration data **410** is modified to reflect the change and a driving application, such as the one described above with respect to methods **200** and **300** of FIGS. **2** and **3**, respectively, interprets the configuration data **410** changes and enforces the changes within the processing container **401**.

[0053] Some processing groups **402** or **403** can be composite processing groups having one or more processing groups

402 and **403** logically associated with one another or integrated with one another. For example, a composite processing group might be a VPN, CDN, forward proxy, transparent proxy, reverse proxy, etc. Additionally, in some embodiments, the some processing groups **402** or **403** may be initially installed and available within the processing container **401**, but may be initially inactive (not available for use by a client). These inactive processing groups **402** or **403** can be dynamically activated based upon a predefined event, such as a license being acquired by the client, a purchase made by the client, or a request for the processing group **402** or **403** being made by the client.

[0054] During operation of the processing system **400**, the processing container **401** defines a processing environment for a client. The client adds, removes, and receives services (processing groups **402** and **403**) via the processing container **401**. Management of the client's processing environment is achieved by managing the configuration data **410** associated with the processing container **401**, the processing groups **402** and **403**, the applications **402**A and **403**A, and the connector interfaces **404**. The client's processing groups **402** and **403** may have applications **402**A and **403**A that process local to or on a client appliance processing device or that process on remote processing devices that are within the client's LAN or external to the client over a WAN.

[0055] FIG. **5** is a diagram of configuration information **500** for defining a client's processing environment. The configuration information **500** is implemented and resides in a machine-accessible and readable medium. In one embodiment, the configuration information **500** is the configuration data described above and consumed within methods **200** and **300** of FIGS. **2** and **3**, respectively, and system **400** of FIG. **4**.

[0056] The configuration information **500** includes processing container definitions **501**, processing group definitions **502**, and connector interface definitions **503**. The container definition **501** defines the rules associated with a processing container. A processing container includes a plurality of processing groups and connector interfaces. The processing groups have applications that combine to provide a service or set of similar services to a client. The connector interfaces are APIs or protocols (e.g., XML, SOAP, RPC, etc.) that permit processing groups to interact with their applications and with other processing groups. The connector interfaces bridge or integrate the processing groups within the processing container.

[0057] The processing group definitions **502** represent a processing context for the processing container. That context supports predefined types of applications and services. In some instances a single processing group definition **502** represents a composite processing group that has two or more processing groups included therein, such as a VPN, CDN, proxy services, etc.

[0058] The connector interface definitions **503** represent APIs or protocols that permit the processing groups to interact with their applications and abstract information or data into generic formats that can be communicated to other processing groups which interact with other applications. Thus, the connector interfaces are communication bridges within the processing container. The applications provided within any particular processing group do not have to be modified to be integrated with the teachings of this invention.

[0059] That is, legacy applications can be easily and seamlessly integrated into processing containers of the present invention. This can be achieved by adding small connector interfaces and connector interface definitions **503** to the configuration information **500**, where these small interfaces use the existing API of a legacy application and abstract information into generic formats for communication to other connector interfaces included within the processing container.

[0060] The configuration information **500** is consumed by a driving application, such as the ones described above with respect to the method **200** and **300** of FIGS. **2** and **3**, respectively, for purposes of establishing a processing environment for a client. The processing environment is managed via the configuration information **500** and can be centrally controlled, and yet, the processing environment is fully distributed over a network and includes a variety of services some of which that are not natively compatible with one another. The new processing environment integrates the services with one another in a plug-and-play fashion and is therefore easily expandable and upgraded as the needs of the client dynamically change.

[0061] Although specific embodiments have been illustrated and described herein, those of ordinary skill in the art will appreciate that any arrangement calculated to achieve the same purpose can be substituted for the specific embodiments shown. This disclosure is intended to cover all adaptations or variations of various embodiments of the invention. It is to be understood that the above description has been made in an illustrative fashion only. Combinations of the above embodiments, and other embodiments not specifically described herein will be apparent to one of ordinary skill in the art upon reviewing the above description. The scope of various embodiments of the invention includes any other applications in which the above structures and methods are used. Therefore, the scope of various embodiments of the invention should be determined with reference to the appended claims, along with the full range of equivalents to which such claims are entitled.

[0062] It is emphasized that the Abstract is provided to comply with 37 C.F.R. §1.72(b), which requires an Abstract that will allow the reader to quickly ascertain the nature and gist of the technical disclosure. It is submitted with the understanding that it will not be used to interpret or limit the scope or meaning of the claims.

[0063] In the foregoing Detailed Description, various features are grouped together in single embodiments for the purpose of description. This method of disclosure is not to be interpreted as reflecting an intention that the claimed embodiments of the invention require more features than are expressly recited in each claim. Rather, as the following claims reflect, inventive subject matter lies in less than all features of a single disclosed embodiment. The following claims are hereby incorporated into the Detailed Description, with each claim standing on its own as a separate preferred embodiment.

What is claimed is:

1. A method for establishing a processing environment, comprising:

configuring processing groups within a processing container, each processing group having a unique context and associated with a particular processing device;

loading applications within each processing group's context for execution on each processing device; and

establishing a processing environment as the processing container that spans the processing groups and their processing devices.

2. The method of claim 1 further comprising, interfacing each of the processing groups with one another within the processing environment.

3. The method of claim 2, wherein interfacing further includes providing a connector interface for each processing group that defines that processing group's context and input and output formats for each of that processing group's applications.

4. The method of claim 3, wherein providing further includes generically defining the input and output formats from a first connector interface to a second connector interface.

5. The method of claim 4 further comprising, receiving, at the first connector interface, instructions for a particular application to process from the second connector interface, the particular application is a legacy application that is unaware of the first and second connector interfaces, and processing, via the first connector interface, the instructions using an application specific format expected by the legacy application.

6. The method of claim 1 further comprising, dynamically adding a new processing group associated with new processing devices and having new applications into the processing environment.

7. The method of claim 1 further comprising, dynamically removing a particular processing group and its processing devices from the processing environment.

8. A method for providing a service to a client, comprising:

configuring a connector interface to translate input and output formats for a service being added to the client;

dynamically adding the service to a particular processing group within a processing environment for the client, the processing environment defined by a processing container and includes multiple other processing groups, each processing group associated with its own processing device and its own processing context;

interfacing the service from the particular processing group to the multiple other processing groups via the connector interface; and

making the service available for access within the processing environment to the client.

9. The method of claim 8 further comprising, authenticating the service before configuring the connector interface.

10. The method of claim 8 wherein configuring further includes identifying the service as a composite service having independent embedded services, each independent embedded service's input and output formats configured within the connector interface.

11. The method of claim 8, wherein dynamically adding further includes selecting the particular processing group based on its processing device being a local device included on the client.

12. The method of claim 8, wherein dynamically adding further includes selecting the particular processing group based on its processing device being external and remote from the client and accessible from a network connection to the client.

13. The method of claim 8, wherein dynamically adding further includes selecting the particular processing group based on its processing device being a composite device distributed over a network connection on multiple devices.

14. The method of claim 8, wherein dynamically adding further includes selecting the particular processing group based on its processing context that is identified as having similar services as the service.

15. The method of claim 8, wherein interfacing further includes using other connector interfaces to communicate with the connector interface, each other connector interface associated with a different processing group within the processing container.

16. A processing system, comprising:

processing devices organized over a network as a processing container that defines a processing environment for a client;

processing groups, each processing group configured as having its own processing context and having its own executing applications and processed on a particular one of the processing devices; and

connector interfaces, each connector interface configured within a particular processing group and processing on that particular processing group's processing device;

wherein a legacy first application processing within a first processing group is configured for interfacing to a second application processing within a second processing group via a first connector interface of the first processing group and a second connector interface of the second processing group.

17. The processing system of claim 16, wherein at least one processing device is a Virtual Machine.

18. The processing system of claim 16, wherein at least one processing group is configured as an authentication service that provides authentication within the processing container.

19. The processing system of claim 18, wherein the processing container is accessible as the processing environment for multiple other clients over the network.

20. The processing system of claim 16, wherein the processing container is configured to permit dynamic addition of new services and new processing groups that span new devices over the network.

* * * * *