(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
5 July 2012 (05.07.2012)

WIPO | PCT

(10) International Publication Number
WO 2012/092113 A2

(51) International Patent Classification:
G06F 21/22 (2006.01) *G06F 9/44* (2006.01)

(21) International Application Number:
PCT/US2011/066790

(22) International Filing Date:
22 December 2011 (22.12.2011)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
12/978,655 27 December 2010 (27.12.2010) US

(71) Applicant *(for all designated States except US)*: **MICROSOFT CORPORATION** [US/US]; One Microsoft Way, Redmond, WA 98052-6399 (US).

(72) Inventors: **MEFFORD, Cread, Wellington, Jr.**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, WA 98052-6399 (US). **BABEY, Matthew, Christopher**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, WA 98052-6399 (US). **CHARDON, Alvin**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, WA 98052-6399 (US). **STEARNS, Scott, Elliot**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, WA 98052-6399 (US). **ANDERSON, Adam, Brady**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, WA 98052-6399 (US). **DE SOUZA, Lidiane, Pereira**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, WA 98052-6399 (US). **ANDERSON, Angela, Mele**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, WA 98052-6399 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

[Continued on next page]

(54) Title: POLICY-BASED ACCESS TO VIRTUALIZED APPLICATIONS

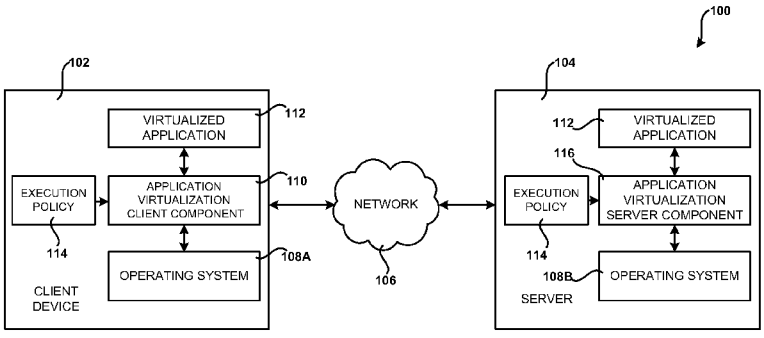

FIG. 1

(57) Abstract: When a request is received to execute a virtualized application, an application virtualization client component evaluates an execution policy to determine if the application may be executed. If the application virtualization client component determines based on the execution policy that the virtualized application may be executed, the application virtualization client component publishes the virtualized application. The application virtualization client component publishes the application by making the virtualized application available for execution if the application is installed, and installing the virtualized application if it is not installed. The application virtualization client component also evaluates the execution policy during execution of the virtualized application. If the application virtualization client component determines that the execution policy is no longer satisfied, the application virtualization client component unpublishes the virtualized application, thereby preventing execution of the virtualized application.

WO 2012/092113 A2

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

POLICY-BASED ACCESS TO VIRTUALIZED APPLICATIONS

BACKGROUND

[0001] Enterprises, governmental agencies, and other types of entities commonly restrict access to certain types of confidential or sensitive information. Restricting access to physical information, such as physical documents, can be easily accomplished using standard access restrictions. Restricting access to digital information, such as documents stored on a server computer, can also be accomplished relatively easily using standard security measures, such as access control lists.

[0002] Restricting the use of computer application programs ("applications") can be more difficult than restricting access to physical items, particularly when a user who is otherwise authorized to use the application must be restricted in some manner. This is especially true for organizations that issue portable computing devices to employees, such as laptop computers, tablet computers, and smartphones. Because employees have continual access to the computing devices that they have been issued, it can be difficult to restrict the use of the applications installed on the computing devices.

[0003] It is with respect to these and other considerations that the disclosure made herein is presented.

SUMMARY

[0004] Technologies are described herein for policy-based access to virtualized applications. Through an implementation of the concepts and technologies presented herein, an application virtualization environment can be provided that is capable of restricting the execution of virtualized applications according to an execution policy. The execution policy may be set by an administrator, and defines the conditions under which virtualized applications may or may not be executed. In this manner, the execution of an application can be restricted based upon various conditions, such as the geographic location of the device upon which the application is executed, the time of day or duration of use of the application, the availability of certain computing resources to the device, or other conditions.

[0005] Embodiments disclosed herein are implemented in conjunction with an application virtualization environment. In particular, an application virtualization client component is configured to provide an environment for execution of a virtualized application. The application virtualization client component also provides functionality for encapsulating the virtualized application from an underlying operating system, other application programs, and system resources. The application virtualization client

component might also provide functionality for loading portions of the virtualized application by streaming needed portions of the virtualized application from an application virtualization server component.

[0006] According to another aspect, the application virtualization client component is configured to provide policy-based access to a virtualized application. In particular, when a request is received to execute a virtualized application, the application virtualization client component evaluates an execution policy to determine if the application may be executed. As discussed briefly above, the execution policy may be set by an administrator, and defines the conditions under which the virtualized application may or may not be executed. For instance, the execution policy may specify that the virtualized application may only be executed when a device executing the application is in a specified geographic area, that the application may only be executed during a specified time of day (e.g. 9 a.m. to 5 p.m.), for a specified duration each day (e.g. one hour per day), or that the application may be executed only when certain computing resources are not limited.

[0007] If the application virtualization client component determines based on the execution policy that the virtualized application may be executed, the application virtualization client component will publish the virtualized application. As utilized herein, the term "publish" means to make a virtualized application available for execution. For instance, in one embodiment, the application virtualization client application publishes the virtualized application by determining if the virtualized application is installed and, if so, makes the virtualized application available for execution. If the virtualized application is not installed, the application virtualization client component may cause the virtualized application to be streamed to the device for execution.

[0008] According to another aspect, the application virtualization client component periodically or continually evaluates the execution policy during execution of the virtualized application. If the application virtualization client component determines that the execution policy is no longer satisfied, the application virtualization client component "unpublishes" the virtualized application. For instance, the application virtualization client component may stop execution of the virtualized application and make the application unavailable for execution. The application virtualization client component might also remove the virtualized application from the client device upon which it is executing.

[0009] According to another aspect, the application virtualization client component also unpublishes the virtualized application if the execution policy cannot be evaluated. For instance, the virtualized application might be unpublished if the execution policy requires that the application only be executed within a particular geographic area, and the geographic area in which the device executing the application cannot be determined. The application virtualization client component might also store audit data regarding disallowed accesses to virtualized applications, the conditions under which applications were unpublished, and other information potentially helpful to an administrator of the application virtualization environment.

[0010] This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended that this Summary be used to limit the scope of the claimed subject matter. Furthermore, the claimed subject matter is not limited to implementations that solve any or all disadvantages noted in any part of this disclosure.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] FIGURE 1 is a software and network architecture diagram showing one illustrative operating environment for the embodiments disclosed herein;

[0012] FIGURES 2A-2B are flow diagrams showing aspects of one illustrative process disclosed herein for policy-based access to virtualized applications, according to one embodiment presented herein; and

[0013] FIGURE 3 is a computer architecture diagram showing an illustrative computer hardware and software architecture for a computing system capable of implementing the various embodiments presented herein.

DETAILED DESCRIPTION

[0014] The following detailed description is directed to technologies for policy-based access to virtualized applications. As discussed briefly above, an application virtualization environment is provided that is configured to enforce restrictions on the execution of a virtualized application based upon an execution policy. In particular, when a request is received to execute a virtualized application, an application virtualization client component evaluates an execution policy to determine if the application may be published and executed. If the application virtualization client component determines based on the execution policy that the virtualized application may be executed, the application virtualization client component publishes and executes the virtualized application.

[0015] The application virtualization client component publishes the application by making the virtualized application available for execution if the application is installed, and installing the virtualized application if it is not installed. The application virtualization client component also evaluates the execution policy during execution of the virtualized application. If the application virtualization client component determines that the execution policy is no longer satisfied, the application virtualization client component unpublishes the virtualized application, thereby preventing execution of the virtualized application. Additional details regarding these and other features will be provided below.

[0016] While the subject matter described herein is presented in the general context of program modules that execute in conjunction with the execution of an operating system and application programs on a computer system, those skilled in the art will recognize that other implementations may be performed in combination with other types of program modules. Generally, program modules include routines, programs, components, data structures, and other types of structures that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the subject matter described herein may be practiced with other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, minicomputers, mainframe computers, and the like.

[0017] In the following detailed description, references are made to the accompanying drawings that form a part hereof, and which are shown by way of illustration specific embodiments or examples. Referring now to the drawings, in which like numerals represent like elements through the several figures, aspects of a computing system and methodology for policy-based access to virtualized applications will be described.

[0018] FIGURE 1 is a software and network architecture diagram showing one illustrative operating environment 100 for the embodiments disclosed herein. The illustrative operating environment 100 shown in FIGURE 1 includes a client device 102 configured to communicate with a server 104 by way of the network 106. The client device 102 is a computing device configured to execute an operating system 108A and an application virtualization client component 110. The client device 102 may be a standard desktop or laptop computer, a tablet computer, a smartphone or any other type of computing device capable of performing the operations presented herein for policy-based

access to virtualized applications. The client device 102 might also be a server computer configured to provide the functionality disclosed herein.

[0019] The server 104 is a computing system configured to execute an operating system 108B and the application virtualization server component 116. It should be appreciated that the server 104 may be a server computer configured to execute the application virtualization server component 110 or may comprise another type of computer system configured to perform the functionality described herein as being performed by the server 104.

[0020] The network 106 illustrated in FIGURE 1 may comprise a wide area or local area network. For instance, the network 106 may be a corporate local area network, a wide area network such as the Internet, or a combination of multiple wide area and local area networks. It should be appreciated that while a single network 106 has been illustrated in FIGURE 1, many other networks may be utilized. It should also be appreciated that while a single client device 102 and server 104 have been illustrated in FIGURE 1, many such devices may be utilized by the embodiments disclosed herein.

[0021] As discussed briefly above, the client device 102 is configured to execute an application virtualization client component 110. The application virtualization client component 110 is a software component configured to provide an application virtualization environment. In this regard, the application virtualization client component 110 is configured to execute a virtualized application 112. The application virtualization client component 110 provides functionality for encapsulating the execution of the virtualized application 112 from the operating system 108A. The application virtualization client component 110 might also provide functionality for encapsulating execution of the virtualized application 112 from other application programs and system resources of the client device 102. For instance, the application virtualization client component 110 might virtualize resources of the operating system 108A or the client device 102. When the virtualized application 112 attempts to access the physical resources, the application virtualization client component 110 presents a virtualized resource to the application 112. In this manner, the virtualized application 112 can be executed in a manner that does not impact the actual resources exposed by the operating system 108A or the client device 102.

[0022] According to other aspects, the application virtualization client component 110 also provides functionality for loading portions of the virtualized application 112 on-demand. In particular, the application virtualization client component 110 may operate in

conjunction with the application virtualization server component 116 to stream needed portions of the virtualized application 112 from the server 104 to the client device 102. In this manner, the virtualized application 112 can be accessed at the client device 102 on-demand. Moreover, because only needed portions of the virtualized application 112 may be streamed from the server 104 to the client device 102, access to the virtualized application 112 may be provided without streaming the entire application 112 from the server 104 to the client device 102.

[0023] Additional details regarding functionalities provided by the application virtualization client component 110 for encapsulating execution of the virtualized application 112 and for streaming the virtualized application 112 from the server 104 to the client device 102 can be found in U.S. Patent No. 7,225,264 filed May 29, 2007 entitled "Systems and Methods for Delivering Content over a Computer Network," U.S. Patent No. 7,200,632 filed April 3, 2007 entitled "Method and System for Serving Software Applications to Client Computers," U.S. Patent No. 7,451,451 filed November 11, 2008 entitled "Operating System Abstraction and Protection Layer," and U.S. Patent No. 7,797,372 filed September 14, 2010 entitled "Serving Software Applications from Servers for Client Computers," each of which is incorporated herein by reference in their entirety.

[0024] The application virtualization client component 110 is also configured to provide policy-based access to the virtualized application 112. In this regard, the server 104 and the client device 102 may store an execution policy 114. The execution policy 114 defines the conditions under which the virtualized application 112 may or may not be executed. For instance, the execution policy 114 may specify that the virtualized application 112 may only be executed when the client device 102 is within a specific geographic area. Alternately, or in combination, the execution policy 114 might also specify that the virtualized application 112 may only be executed during a specified time of day or a specified duration of time each day.

[0025] As another example, the execution policy 114 may specify that the virtualized application 112 may only be executed when certain computing resources of the client device 102 are not limited. For instance, if a certain amount of central processing unit or network bandwidth is unavailable to the client device 102, the execution policy 114 may specify that the virtualized application 112 cannot be executed. It should be appreciated that the execution policy 114 might specify other restrictions based on

physical location, time, or other factors. It should also be appreciated that an administrator of the client device 102 or the server 104 may set the execution policy 114.

[0026] When a request is received at the client device 102 to execute the virtualized application 112, the application virtualization client component 110 is configured to evaluate the execution policy 114 to determine if the application 112 may be executed. In order to evaluate the execution policy 114, the application virtualization client component 110 may operate with other components in order to obtain data necessary to evaluate the execution policy 114. For instance, the application virtualization client component 110 may operate in conjunction with a software or hardware component capable of determining the physical location of the client device 102.

[0027] Illustrative technologies for identifying the location of the client device 102 include cellular triangulation, global positioning system ("GPS") location, A-GPS location, wireless signal strength-based location, wired signal strength-based location, Internet protocol address-based location determination, and others. The application virtualization client component 110 might also operate in conjunction with other types of components in order to obtain data necessary to evaluate the execution policy 114. As will be described in greater detail below, if the application virtualization client component 110 cannot obtain data necessary to evaluate the execution policy 114, execution of the virtualized application 112 may be prohibited.

[0028] It should be appreciated that, according to embodiments, evaluation of the execution policy 114 may occur at the client device 102, the server 104, or a combination of the client device 102 and the server 104. For instance, in one implementation, the application virtualization client component 110 is configured to request evaluation of the execution policy 114 by the application virtualization server component 116. If the application virtualization server component 116 is unavailable, the application virtualization client component 110 may evaluate the execution policy 114 at the client device 102. Alternately, if the application virtualization server component 116 is unavailable to evaluate the execution policy 114, the application virtualization client component 110 may prohibit execution of virtualized application 112. Alternately, the application virtualization client component 110 may be configured to evaluate the execution policy 114 at the client device 102 without any assistance from the server 104.

[0029] If the application virtualization client component 110 determines based on the execution policy 114 that the virtualized application 112 may be executed, the application virtualization client component 110 will publish the virtualized application 112

for use. As described briefly above, the term "publish" means to make the virtualized application 112 available for execution on the client device 102. For instance, in one embodiment, the application virtualization client component 110 publishes the virtualized application 112 by determining if the virtualized application 112 is installed at the client device 102. If the virtualized application 112 is installed at the client device 102, the application virtualization client component 110 makes the virtualized application 112 available for execution. If the virtualized application 112 is not installed at the client device 102, the application virtualization client component 110 may cause the virtualized application 112 to be streamed from the server 104 to the client device 102 for execution.

[0030] The application virtualization client component 110 is also configured to evaluate the execution policy 114 during execution of the virtualized application 112. For instance, the application virtualization client component 110 may periodically or continually evaluate the execution policy 114 to ensure that the execution policy 114 is being met. If the application virtualization client component 110 determines that the execution policy 114 is no longer satisfied, the application virtualization client component 110 unpublishes the virtualized application 112. For instance, the application virtualization client component 110 may stop execution of the virtualized application 112 and make the application 112 unavailable for execution. The application virtualization client component 110 might also remove the virtualized application 112 from the client device 102. If the execution policy 114 is satisfied again later, the application virtualization client component 110 may republish the virtualized application 112 by streaming the virtualized application 112 from the server 104 to the client device 102.

[0031] As discussed briefly above, the application virtualization client component 110 also unpublishes the virtualized application 112 if the execution policy 114 cannot be evaluated. For instance, the virtualized application 112 may be unpublished if the execution policy 114 requires that the application 112 only be executed within a particular geographic area and the geographic area cannot be determined by the application virtualization client component 110. If the client device 102 is later returned to the geographic area in which execution of the application 112 is permitted, the application virtualization client component 110 may republish the application 112 for execution.

[0032] According to other aspects, the application virtualization client component 110 is configured to store audit data for use by an administrator of the server 104 and the client device 102. For instance, the audit data might include data identifying disallowed accesses to the virtualized application 112, the conditions under which the application

virtualization client component 110 unpublished the application 112, and other information potentially helpful to an administrator of the application virtualization environment provided by the components 110 and 116. Additional details regarding the operation of the application virtualization client component 110, the client device 102, and the server 104 will be provided below with respect to FIGURES 2A-2B.

[0033] FIGURES 2A-2B are flow diagrams showing a routine 200 that illustrates aspects of one illustrative process disclosed herein for policy-based access to virtualized applications. It should be appreciated that the logical operations described herein with respect to FIGURES 2A-2B and the other FIGURES are implemented (1) as a sequence of computer implemented acts or program modules running on a computing system and/or (2) as interconnected machine logic circuits or circuit modules within the computing system. The implementation is a matter of choice dependent on the performance and other requirements of the computing system. Accordingly, the logical operations described herein are referred to variously as operations, structural devices, acts, or modules. These operations, structural devices, acts and modules may be implemented in software, in firmware, in special purpose digital logic, and any combination thereof. It should also be appreciated that more or fewer operations may be performed than shown in the figures and described herein. These operations may also be performed in a different order than those described herein.

[0034] The routine 200 begins at operation 202, where the application virtualization client component 110 determines whether a request has been received to execute the virtualized application 112. For instance, a user of the client device 102 may make a request to execute the application 112. If no such request has been received, the routine 200 proceeds back to operation 202 where another such determination is made. If the application virtualization client component 110 determines that a request has been received to execute the virtualized application 112, the routine 200 proceeds from operation 202 to operation 204.

[0035] At operation 204, the application virtualization client component 110 attempts to evaluate the execution policy 114. As discussed above, the application virtualization client component 110 may utilize functionality provided by other components to obtain data necessary to evaluate the execution policy 114. For instance, the application virtualization client component 110 may obtain data identifying the geographic location of the client device 102, the time of day, or other information

necessary to evaluate the execution policy 114. From operation 204, the routine 200 proceeds to operation 206.

[0036] At operation 206, the application virtualization client component 110 determines whether it was capable of evaluating the execution policy 114. For instance, the application virtualization client component 110 may be unable to evaluate the execution policy 114 if data, such as the geographic location of the client device 102, is unavailable. The application virtualization client component 110 might also be unable to evaluate the execution policy 114 in other embodiments if a network connection cannot be made to the server 104. Other types of factors might also prevent the application virtualization client component 110 from evaluating the execution policy 114.

[0037] If the execution policy 114 cannot be evaluated, the routine 200 proceeds from operation 206 to operation 208. At operation 208, the application virtualization client component 110 denies the request to execute the virtualized application 112. Additionally, as discussed above, the application virtualization client component 110 may store audit data regarding the disallowed access to the virtualized application 112, along with other information potentially helpful to an administrator of the application virtualization environment. From operation 208, the routine 200 proceeds to operation 202, described above, where another request to execute the application 112 may be processed.

[0038] If the execution policy 114 was evaluated by the application virtualization client component 110, the routine 200 proceeds from operation 206 to operation 210. At operation 210 a determination is made as to whether the execution policy 114 was satisfied. If the execution policy 114 was not satisfied, the routine 200 proceeds from operation 210 to operation 208 where the request to execute the virtualized application 112 is denied. Additionally, as described above, audit data might also be stored.

[0039] If the execution policy 114 was satisfied, the routine 200 proceeds to operation 212 where the application virtualization client component 110 publishes the virtualized application 112. As discussed above, in order to publish the application 112, the application virtualization client component 110 may determine if the application 112 is installed at the client device 102. If the application 112 is installed, the application virtualization client component 110 may make the virtualized application available for execution for execution on the client device 102. If the virtualized application 112 is not installed at the client device 102, the application virtualization client component 110 may cause the virtualized application 112 to be streamed from the server 104 to the client

device 102 for execution. Once the application 112 has been published, the routine 200 proceeds from operation 212 to operation 214.

[0040] According to one implementation, the application virtualization client component 110 may be configured to store the state of the virtualized application 112 each time the application 112 is unpublished. For instance, the application virtualization client component 110 may store the contents of memory, registers, and data describing the state of resources of the client device 102 each time the application 112 is unpublished. In this implementation, the application virtualization client component 110 may be configured to restore a previously saved state of the virtualized application 112 each time the application 112 is republished. For instance, at operation 214 of the routine 200, the application virtualization client component 110 restores the previously saved state of the virtualized application 112. In this manner, a user of the client device 102 may be presented with a virtualized application 112 upon republication that is in the same state as it was at the time the application 112 was unpublished.

[0041] From operation 214, the routine 200 proceeds to operation 216 where the application virtualization client component 110 causes the virtualized application 112 to be executed. As discussed above, the application virtualization client component 110 provides functionality for encapsulating the execution of the virtualized application 112 from the operating system 108, other application programs, and system resources. From operation 216, the routine 200 proceeds to operation 218.

[0042] At operation 218 the application virtualization client component 110 attempts to evaluate the execution policy 114 during execution of the virtualized application 112. In this manner, the application virtualization client component 110 can enforce restrictions imposed by the execution policy 114 even while the virtualized application 112 is executing.

[0043] From operation 218, the routine 200 proceeds to operation 220 where the application virtualization client component 110 determines whether it was able to evaluate the execution policy 114. For instance, as discussed above, the application virtualization client component 110 may be unable to evaluate the execution policy 114 if needed data is unable or if a connection cannot be established to the server 104.

[0044] If the execution policy 114 could be evaluated, the routine 200 proceeds from operation 220 to operation 222. At operation 222 the application virtualization client component 110 determines whether the execution policy 114 is satisfied. If so, the routine 200 proceeds from operation 222 to operation 216, described above, where execution of

the virtualized application 112 continues. If the execution policy 114 is not satisfied, the routine 200 proceeds from operation 222 to operation 224. Additionally, the routine 200 proceeds from operation 220 to operation 224 if the application virtualization client component 110 is unable to evaluate the execution policy 114.

[0045] At operation 224, the application virtualization client component 110 stores the state of the virtualized application 112. As discussed above, the application virtualization client component 110 may store memory, register contents, and other information necessary to return the virtualized application 112 to its current state after republication. Once the state of the virtualized application 112 has been stored, the routine 200 proceeds to operation 226.

[0046] At operation 226 the application virtualization client component 110 unpublishes the virtualized application 112. As discussed briefly above, the application virtualization client component 110 may unpublish the virtualized application 112 by making the application 112 unavailable for execution of the client device 102. The application virtualization client component 110 might also remove the virtualized application 112 and its state from the client device 102. The virtualized application 112 may be streamed from the server 104 to the client device 102 the next time the virtualized application 112 is executed.

[0047] The application virtualization client component 110 might also store audit data at operation 226 regarding the disallowed access to the application 112, the conditions under which the application 112 was unpublished, and other information potentially helpful to an administrator of the client device 102 and the server 104. From operation 226, the routine 200 proceeds to operation 202, described above, where additional requests to the application 112 are processed in the manner described above.

[0048] FIGURE 3 is a computer architecture diagram showing an illustrative computer hardware and software architecture for a computing system capable of implementing the various embodiments presented herein. The computer architecture shown in FIGURE 3 illustrates a conventional desktop, laptop computer, or server computer and may be utilized to execute the various software components described herein.

[0049] The computer architecture shown in FIGURE 3 includes a central processing unit 302 ("CPU"), a system memory 308, including a random access memory 314 ("RAM") and a read-only memory ("ROM") 316, and a system bus 304 that couples the memory to the CPU 302. A basic input/output system ("BIOS") containing

the basic routines that help to transfer information between elements within the computer 300, such as during startup, is stored in the ROM 316. The computer 300 further includes a mass storage device 310 for storing an operating system 318, application programs, and other program modules, which will be described in greater detail below.

[0050] The mass storage device 310 is connected to the CPU 302 through a mass storage controller (not shown) connected to the bus 304. The mass storage device 310 and its associated computer-readable storage media provide non-volatile storage for the computer 300. Although the description of computer-readable media contained herein refers to a mass storage device, such as a hard disk or CD-ROM drive, it should be appreciated by those skilled in the art that computer-readable storage media can be any available computer storage media that can be accessed by the computer 300.

[0051] By way of example, and not limitation, computer-readable storage media may include volatile and non-volatile, removable and non-removable media implemented in any method or technology for storage of information such as computer-readable instructions, data structures, program modules or other data. For example, computer-readable storage media includes, but is not limited to, RAM, ROM, EPROM, EEPROM, flash memory or other solid state memory technology, CD-ROM, digital versatile disks ("DVD"), HD-DVD, BLU-RAY, or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other non-transitory medium which can be used to store the desired information and which can be accessed by the computer 300.

[0052] It should be appreciated that the computer-readable media disclosed herein also encompasses communication media. Communication media typically embodies computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of the any of the above should also be included within the scope of computer readable media. Computer-readable storage media does not encompass communication media.

[0053] According to various embodiments, the computer 300 may operate in a networked environment using logical connections to remote computers through a network such as the network 320. The computer 300 may connect to the network 320 through a network interface unit 306 connected to the bus 304. It should be appreciated that the network interface unit 306 may also be utilized to connect to other types of networks and remote computer systems. The computer 300 may also include an input/output controller 312 for receiving and processing input from a number of other devices, including a keyboard, mouse, or electronic stylus (not shown in FIGURE 3). Similarly, an input/output controller may provide output to a display screen, a printer, or other type of output device (also not shown in FIGURE 3).

[0054] As mentioned briefly above, a number of program modules and data files may be stored in the mass storage device 310 and RAM 314 of the computer 300, including an operating system 318 suitable for controlling the operation of a networked desktop, laptop, or server computer. The mass storage device 310 and RAM 314 may also store one or more program modules. In particular, the mass storage device 310 and the RAM 314 may store the virtualized application 112, the application virtualization client component 110, and/or the other software components described above. The mass storage device 310 and RAM 314 may also store other program modules and data, such as the execution policy 114.

[0055] In general, software applications or modules may, when loaded into the CPU 302 and executed, transform the CPU 302 and the overall computer 300 from a general-purpose computing system into a special-purpose computing system customized to perform the functionality presented herein. The CPU 302 may be constructed from any number of transistors or other discrete circuit elements, which may individually or collectively assume any number of states. More specifically, the CPU 302 may operate as one or more finite-state machines, in response to executable instructions contained within the software or modules. These computer-executable instructions may transform the CPU 302 by specifying how the CPU 302 transitions between states, thereby physically transforming the transistors or other discrete hardware elements constituting the CPU 302.

[0056] Encoding the software or modules onto a mass storage device may also transform the physical structure of the mass storage device or associated computer readable storage media. The specific transformation of physical structure may depend on various factors, in different implementations of this description. Examples of such factors may include, but are not limited to: the technology used to implement the computer

readable storage media, whether the computer readable storage media are characterized as primary or secondary storage, and the like. For example, if the computer readable storage media is implemented as semiconductor-based memory, the software or modules may transform the physical state of the semiconductor memory, when the software is encoded therein. For example, the software may transform the states of transistors, capacitors, or other discrete circuit elements constituting the semiconductor memory.

[0057] As another example, the computer readable storage media may be implemented using magnetic or optical technology. In such implementations, the software or modules may transform the physical state of magnetic or optical media, when the software is encoded therein. These transformations may include altering the magnetic characteristics of particular locations within given magnetic media. These transformations may also include altering the physical features or characteristics of particular locations within given optical media, to change the optical characteristics of those locations. Other transformations of physical media are possible without departing from the scope and spirit of the present description, with the foregoing examples provided only to facilitate this discussion.

[0058] Based on the foregoing, it should be appreciated that technologies for policy-based access to virtualized applications have been presented herein. Although the subject matter presented herein has been described in language specific to computer structural features, methodological acts, and computer readable media, it is to be understood that the invention defined in the appended claims is not necessarily limited to the specific features, acts, or media described herein. Rather, the specific features, acts and mediums are disclosed as example forms of implementing the claims.

[0059] The subject matter described above is provided by way of illustration only and should not be construed as limiting. Various modifications and changes may be made to the subject matter described herein without following the example embodiments and applications illustrated and described, and without departing from the true spirit and scope of the present invention, which is set forth in the following claims.

What is claimed is:

1. A computer-implemented method comprising performing computer-implemented operations for:

receiving a request to execute a virtualized application;

in response to receiving the request, evaluating an execution policy for the virtualized application to determine if the virtualized application may be executed; and

publishing the virtualized application in response to determining that the virtualized application may be executed.

2. The computer-implemented method of claim 1, wherein the execution policy specifies that the virtualized application may only be executed when a device executing the virtualized application is located in a specified geographical location.

3. The computer-implemented method of claim 1, wherein the execution policy specifies that the virtualized application may only be executed during a specified period of time.

4. The computer-implemented method of claim 1, wherein the execution policy specifies that the virtualized application may only be executed for a specified duration.

5. The computer-implemented method of claim 1, wherein the execution policy specifies that the virtualized application may only be executed when computing resources utilized by a device executing the virtualized application are not limited.

6. The computer-implemented method of claim 1, further comprising:

determining whether the execution policy can be evaluated; and

unpublishing the virtualized application in response to determining that the execution policy cannot be evaluated.

7. The computer-implemented method of claim 6, wherein publishing the virtualized application comprises:

determining whether the virtualized application is installed at a device;

making the virtualized application available for execution if the virtualized application is installed at the device; and

streaming the virtualized application to the device if the virtualized application is not installed at the device.

8. A computer-readable storage medium having computer-executable instructions stored thereupon which, when executed by a computer, cause the computer to:

receive a request to execute a virtualized application;

evaluate an execution policy for the virtualized application to determine if the virtualized application may be executed in response to receiving the request; and

publish the virtualized application in response to determining that the virtualized application may be executed.

9. The computer-readable storage medium of claim 8, having further computer-executable instructions stored thereon which, when executed by the computer, cause the computer to:

evaluate the execution policy during execution of the virtualized application to determine if the virtualized application may continue to be executed; and to

unpublish the virtualized application in response to determining that the virtualized application may not continue to be executed.

10. The computer-readable storage medium of claim 9, having further computer-executable instructions stored thereon which, when executed by the computer, cause the computer to:

determine whether the execution policy can be evaluated; and

unpublish the virtualized application in response to determining that the execution policy cannot be evaluated.

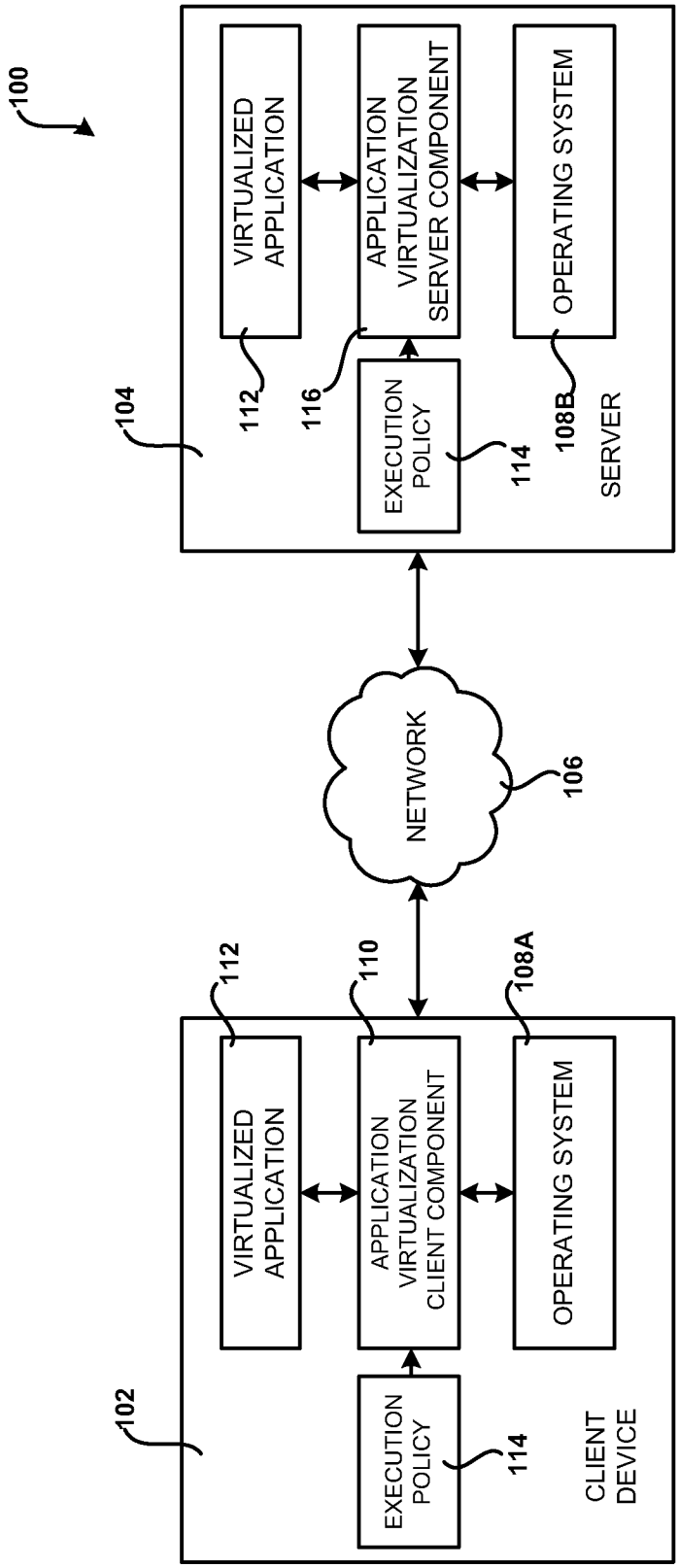
100
104
VIRTUALIZED APPLICATION
112
APPLICATION VIRTUALIZATION SERVER COMPONENT
116
OPERATING SYSTEM
108B
SERVER
EXECUTION POLICY
114
NETWORK
106
102
VIRTUALIZED APPLICATION
112
APPLICATION VIRTUALIZATION CLIENT COMPONENT
110
OPERATING SYSTEM
108A
CLIENT DEVICE
EXECUTION POLICY
114

FIG. 1

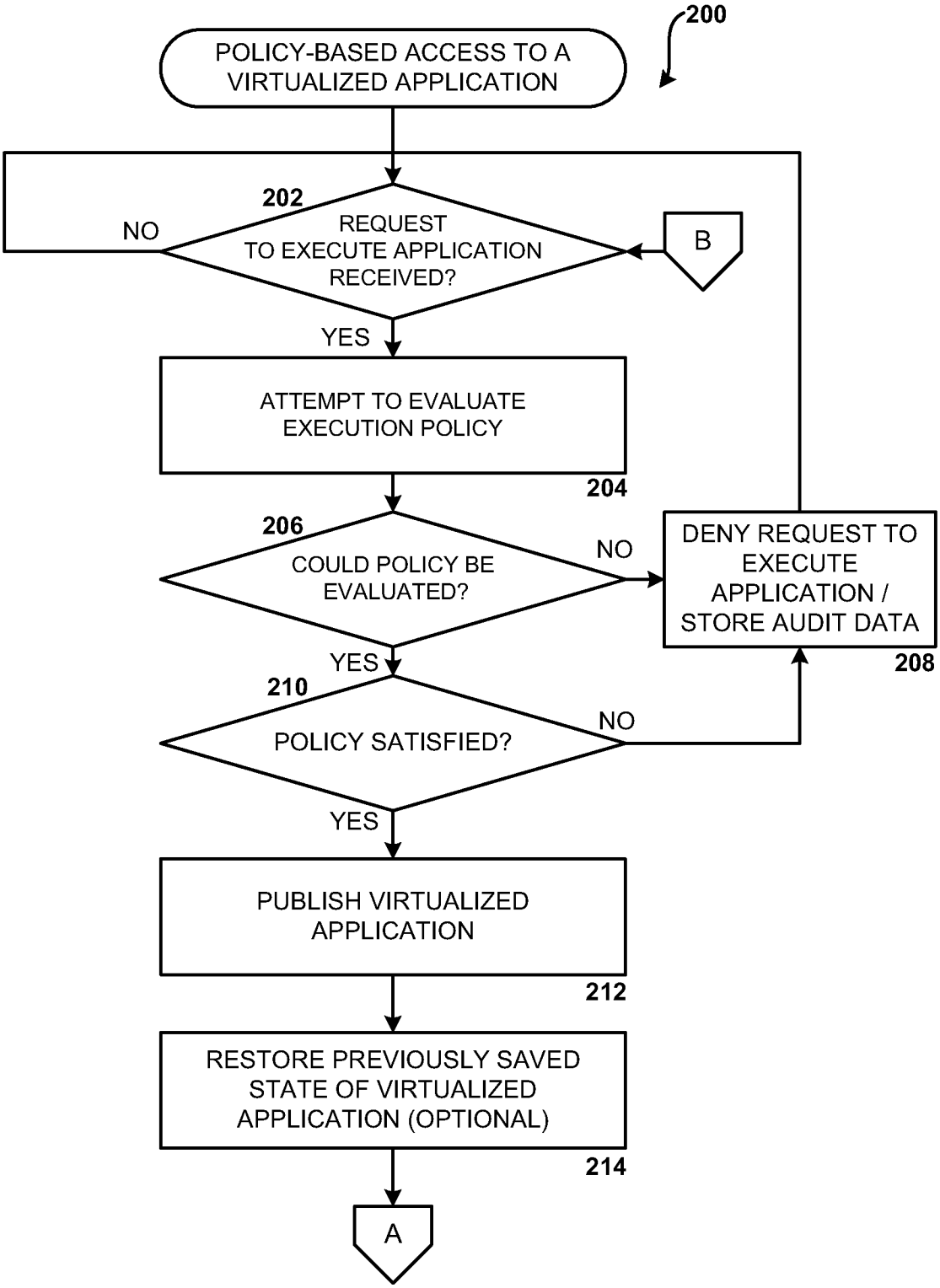
200
POLICY-BASED ACCESS TO A VIRTUALIZED APPLICATION
202
REQUEST TO EXECUTE APPLICATION RECEIVED?
NO
B
YES
ATTEMPT TO EVALUATE EXECUTION POLICY
204
206
COULD POLICY BE EVALUATED?
NO
DENY REQUEST TO EXECUTE APPLICATION / STORE AUDIT DATA
208
YES
210
POLICY SATISFIED?
NO
YES
PUBLISH VIRTUALIZED APPLICATION
212
RESTORE PREVIOUSLY SAVED STATE OF VIRTUALIZED APPLICATION (OPTIONAL)
214
A

FIG. 2A

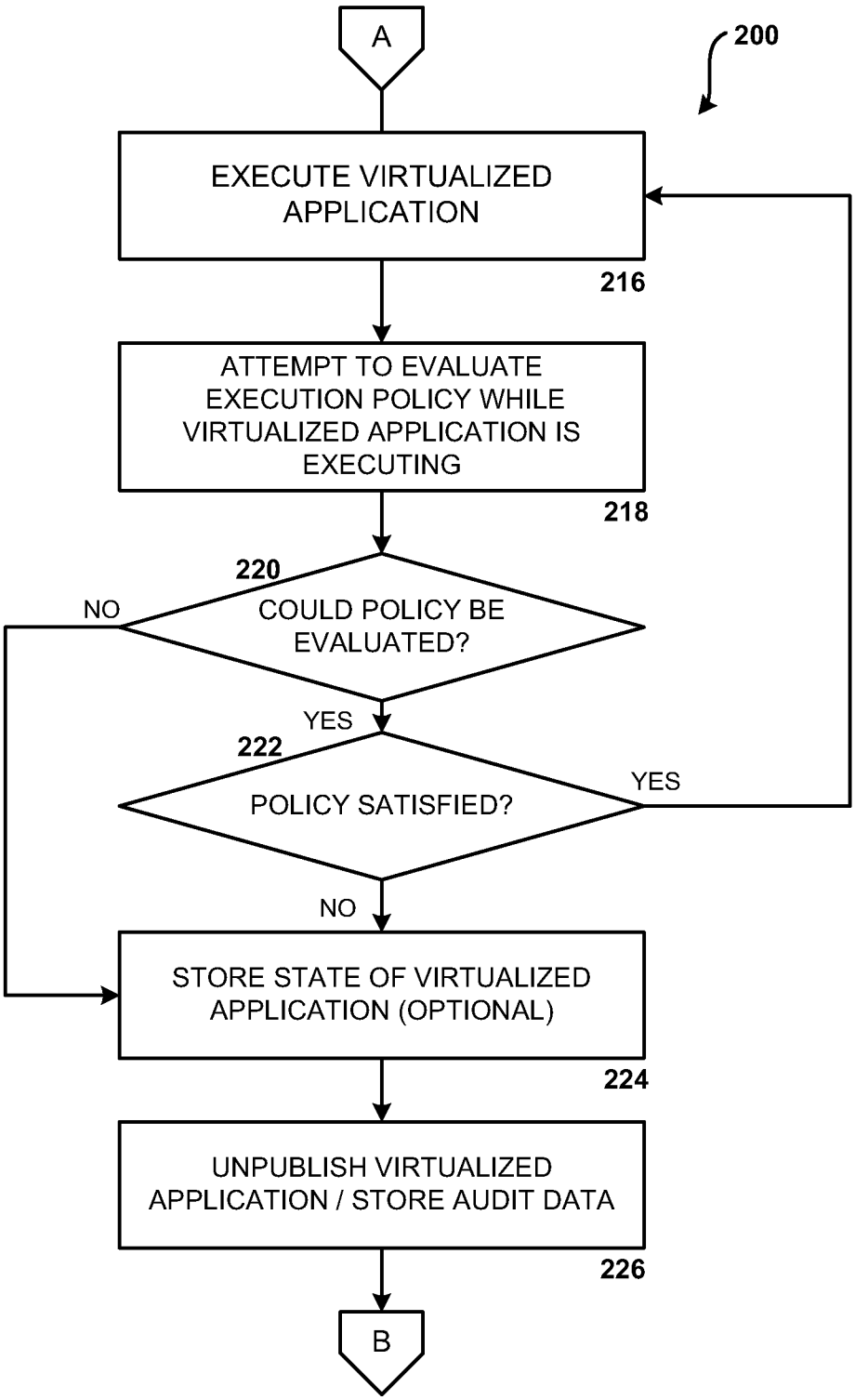
A
200
EXECUTE VIRTUALIZED APPLICATION
216
ATTEMPT TO EVALUATE EXECUTION POLICY WHILE VIRTUALIZED APPLICATION IS EXECUTING
218
220
COULD POLICY BE EVALUATED?
NO
YES
222
POLICY SATISFIED?
YES
NO
STORE STATE OF VIRTUALIZED APPLICATION (OPTIONAL)
224
UNPUBLISH VIRTUALIZED APPLICATION / STORE AUDIT DATA
226
B

FIG. 2B

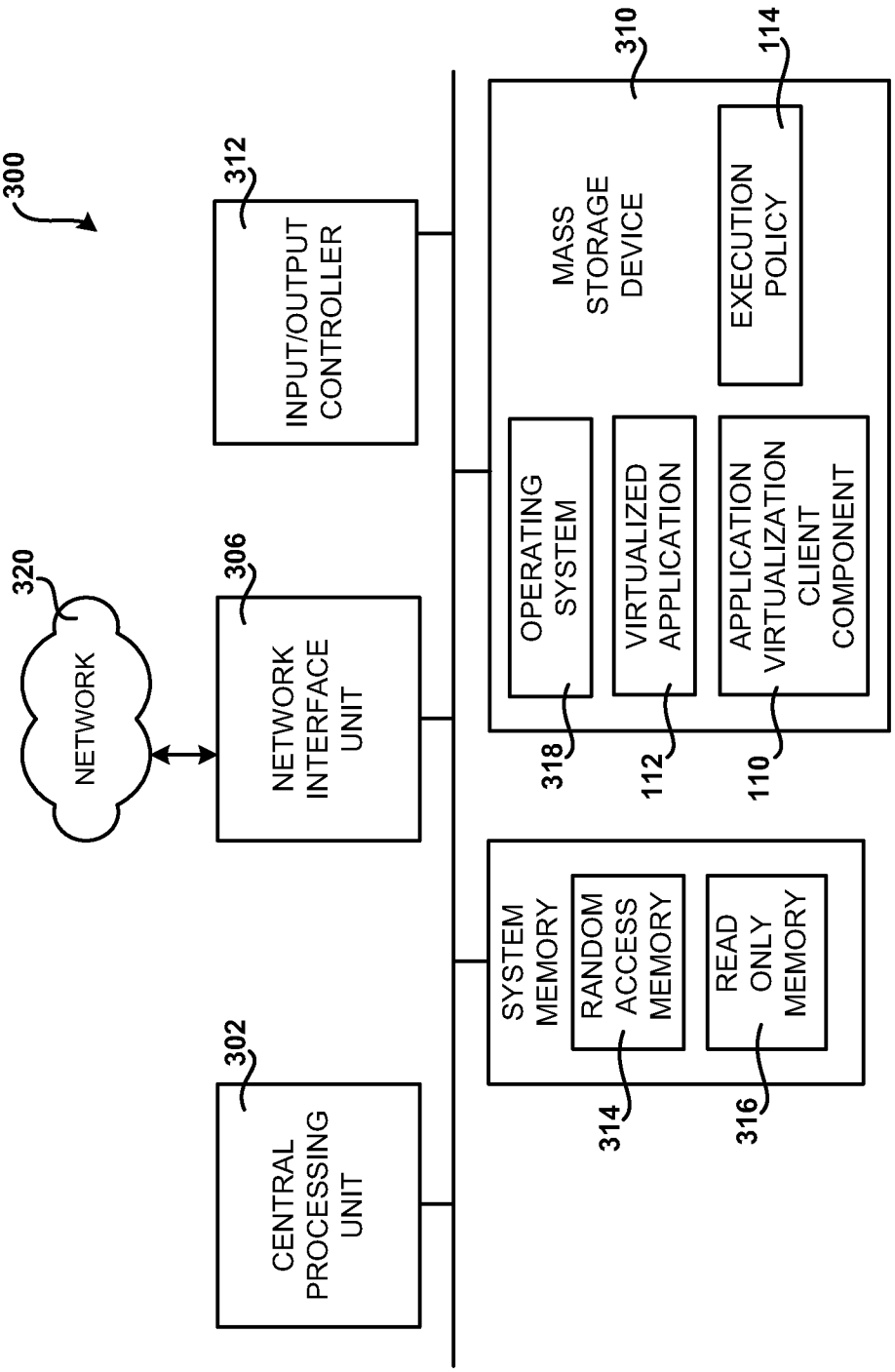
300
312
INPUT/OUTPUT CONTROLLER
310
MASS STORAGE DEVICE
114
EXECUTION POLICY
OPERATING SYSTEM
VIRTUALIZED APPLICATION
APPLICATION VIRTUALIZATION CLIENT COMPONENT
318
112
110
320
NETWORK
306
NETWORK INTERFACE UNIT
SYSTEM MEMORY
RANDOM ACCESS MEMORY
READ ONLY MEMORY
314
316
302
CENTRAL PROCESSING UNIT

FIG. 3