(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) **EP 3 834 099 B9**

(12) **CORRECTED EUROPEAN PATENT SPECIFICATION**

(15) Correction information:
Corrected version no 1 (W1 B1)
Corrections, see
Claims EN 11

(48) Corrigendum issued on:
03.07.2024 Bulletin 2024/27

(45) Date of publication and mention of the grant of the patent:
20.03.2024 Bulletin 2024/12

(21) Application number: **19727791.6**

(22) Date of filing: **13.05.2019**

(51) International Patent Classification (IPC):
G06F 16/901 *(2019.01)* ***G06F 16/904*** *(2019.01)*

(52) Cooperative Patent Classification (CPC):
G06F 16/9027; G06F 9/451; G06F 16/904

(86) International application number:
PCT/US2019/031939

(87) International publication number:
WO 2020/033023 (13.02.2020 Gazette 2020/07)

(54) **ABILITY TO BROWSE AND RANDOMLY ACCESS A LARGE HIERARCHY IN NEAR CONSTANT TIME IN A STATELESS APPLICATION**

FÄHIGKEIT ZUM DURCHSUCHEN UND DIREKTEN ZUGREIFEN AUF EINE GROSSE HIERARCHIE IN NAHEZU KONSTANTER ZEIT IN EINER ZUSTANDSLOSEN ANWENDUNG

CAPACITÉ DE NAVIGATION ET D'ACCÈS ALÉATOIRE À UNE GRANDE HIÉRARCHIE EN TEMPS PRESQUE CONSTANT DANS UNE APPLICATION SANS ÉTAT

(84) Designated Contracting States:
AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR

(30) Priority: **07.08.2018 US 201862715563 P**
27.03.2019 US 201916367087

(43) Date of publication of application:
16.06.2021 Bulletin 2021/24

(73) Proprietor: **ORACLE INTERNATIONAL CORPORATION**
Redwood Shores, California 94065 (US)

(72) Inventors:
- **GARG, Anurag**
  Redwood Shores, California 94065 (US)
- **WOODARD, Jeffrey Leon**
  Redwood Shores, California 94065 (US)

(74) Representative: **Dendorfer, Claus et al**
Dendorfer & Herrmann
Patentanwälte Partnerschaft mbB
Neuhauser Straße 47
80331 München (DE)

(56) References cited:
US-A1- 2016 321 287

Note: Within nine months of the publication of the mention of the grant of the European patent in the European Patent Bulletin, any person may give notice to the European Patent Office of opposition to that patent, in accordance with the Implementing Regulations. Notice of opposition shall not be deemed to have been filed until the opposition fee has been paid. (Art. 99(1) European Patent Convention).

EP 3 834 099 B9

Description

TECHNICAL FIELD

[0001] The present disclosure relates to various techniques for browsing hierarchical data. Specifically, this disclosure is directed to efficiently improving response times for displaying nodes from a very large hierarchy of nodes.

BACKGROUND

[0002] There are several approaches for browsing data maintained by a remote machine. When browsing small sets of remote data, the browser may be able to download and cache the entire dataset at the beginning of browsing so that no further data retrieval is necessary during the browsing session. However, with datasets having on the order of a million entries, the entire dataset may be too large to fit in a local cache. Even if the dataset fits in a local cache, retrieving the entire data set when only a small portion of the dataset is to be accessed may be inefficient. A common approach of loading data only as the user scrolls down doesn't allow any area of the hierarchy to be displayed without first loading all prior nodes.

[0003] Fetching and cache management strategies may be used to access data stored locally that has been retrieved from remotely stored datasets. In one example, a browser provides a scrolling interface for browsing "linear" data. "Linear data" as used herein refers to data items that have a relative ordering (such as chronological, alphabetical, or numerical). The relative ordering indicates which data items are immediately next to other data items. The lists of data are displayed according to that relative order.

[0004] The approaches described in this section are approaches that could be pursued, but not necessarily approaches that have been previously conceived or pursued. Therefore, unless otherwise indicated, it should not be assumed that any of the approaches described in this section qualify as prior art merely by virtue of their inclusion in this section.

[0005] US 2016/0321287 A1 discloses techniques for fast views of items in file directories or file folders when interacting with a cloud-based service platform. A first example concerns techniques for prefetching metadata. When a user seeks to navigate to a certain directory, metadata is fetched for the certain directory. Without requiring an explicit call, additional metadata for additional directories is automatically prefetched, based on the anticipation that, if a given directory is opened, then there is a good chance that subdirectories for that directory will also be opened in the near future. A second example concerns file tree streaming, which is a technique for progressively building a local file tree view of shared content. A virtual file system may receive a request from a file tree viewing application to view a certain file tree item. This request is serviced by means of a progressive file tree build, which can be stopped at any time, even before an entire hierarchical level has been retrieved.

SUMMARY

[0006] The present invention is defined by the independent claims. The dependent claims concern optional features of some embodiments of the invention. Due account shall be taken of any element which is equivalent to an element specified in the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] The embodiments are illustrated by way of example and not by way of limitation in the figures of the accompanying drawings. It should be noted that references to "an" or "one" embodiment in this disclosure are not necessarily to the same embodiment, and they mean at least one. In the drawings:

Figure 1A is a block diagram that illustrates components of the system, in accordance with one or more embodiments;

Figure 1B illustrates Hierarchy of Nodes 122 in more detail, in accordance with one or more embodiments;

Figure 1C illustrates a local cache of nodes corresponding to nodes in the hierarchical tree of data illustrated in Figure 1B, in accordance with one or more embodiments;

Figure 2 is a flow diagram that illustrates retrieving and displaying a target node, in accordance with one or more embodiments; and

Figure 3 shows a block diagram that illustrates a computer system in accordance with one or more embodiments.

DETAILED DESCRIPTION

[0008] In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding.

1. GENERAL OVERVIEW

[0009] A large hierarchical data set may be visually represented by thousands or even millions of nodes. A system, displaying a visual representation (e.g., nodes) corresponding to the hierarchical data, is implemented remotely from a datastore that includes the hierarchical data. One or more embodiments aim to present requested nodes corresponding to a subset of the hierarchical data without any significant lag time, as though the hierarchical data is locally stored even though the hierarchical data is remotely stored.

[0010] The system fetches nodes that are to be displayed rather than fetching all nodes representing the entirety of the hierarchical data. As an example, an in-

terface displays a section of the visual representation and allows a user to scroll toward a particular non-displayed section of the visual representation. Based on the scrolling, the particular non-displayed section corresponds to nodes that are now to be displayed. As another example, an interface allows a user to identify an individual node (i.e., by random access) in the hierarchy that is not shown in the display and not within reasonable scrolling distance of the currently-displayed data. Upon receiving the user-specified node, the specified node and nodes related to the specified node are displayed.

[0011] Subsequent to and responsive to detecting user input that causes nodes to be displayed that are not currently displayed, the system fetches the nodes corresponding to the particular non-displayed section without fetching nodes corresponding to the entirety of the hierarchical data. Identifying and fetching a subset of nodes to be displayed is quick in comparison to fetching all the nodes of the visual representation corresponding to the entirety of the hierarchical data. The speed at which the subset of nodes is fetched and displayed may be sufficiently similar from a user perspective to the speed at which locally stored nodes are displayed in response to a user request.

[0012] In addition to fetching the nodes to be displayed, one or more embodiments pre-fetch additional nodes that are related to and/or adjacent to the set of nodes to be displayed

[0013] While the initial user input does not result in display of the additional nodes, further user input (e.g., scrolling from the initial display of the particular node and the set of nodes displayed with the particular node) may result in display of the additional adjacent nodes. Fetching the additional adjacent nodes without fetching the entire set of nodes is efficient while allowing for, if requested, a quick presentation of the additional adjacent nodes.

[0014] Some embodiments described in this Specification and/or recited in the claims may not be included in this General Overview section.

2. ARCHITECTURE FOR BROWSING HIERARCHICAL DATA

[0015] A tree browser may allow a user to browse hierarchical data. In an embodiment in which nodes from a hierarchy are displayed in an ordered sequence in a display window, the ordered sequence is determined by traversing the hierarchy in a particular search order. For example, in an embodiment the ordered sequence may be generated by following these rules starting with a particular node A in the hierarchy:

(a) start with an empty sequence, and start at a particular node in the hierarchy, such as node A.
(b) add the particular node to the sequence.
(c) if the particular node has one or more child nodes and the particular node is in an expanded state then:
(d) perform steps (b) and (c) for each of the child nodes of the particular node in order of the child nodes. (This is a recursive algorithm in which a child node becomes the particular node when performing steps b and c).

[0016] The ordering produced by these rules results in a linearization of the hierarchy similar to the way many file system contents are displayed to a user, with the containing folder or directory first, then a list of contents in the directory including files and other directories.

[0017] The nodes in the ordered sequence may be affected by an expansion status of a node. When a parent node is expanded, the child nodes of that parent node appear in the ordered sequence of nodes, and when the parent node is collapsed, the child nodes do not appear.

[0018] In addition to scrolling through an ordered list of data representing a hierarchy (tree) of nodes, a tree browser may also allow for a user to expand or collapse a node of the tree. When a user expands a node in the display, the display changes by inserting child nodes into the ordered list shown in the display. Inserting nodes in the middle of the displayed list may push nodes, appearing toward the bottom of the list, off the bottom end of the display. The position of a node displayed to a user relative to other nodes on a screen may change based on the expansion of nodes in the hierarchy. Thus, the set of nodes to be displayed depends on the expansion state of nodes related to the particular node.

[0019] A hierarchical data browser may fetch data from a remote machine that includes all the nodes in a subtree of the hierarchy, then only display nodes requested by the user. That is, a fully expanded subtree may be fetched.

[0020] Figure 1A is a block diagram that illustrates components of the system, in accordance with one or more embodiments. Components may be omitted, modified, rearranged, combined, or otherwise modified.

[0021] Hierarchy of Nodes 122 represents a very large dataset of hierarchical data that is browsed. The nodes in the hierarchy are sent upon request to a Local Machine 142 that hosts users browsing the data. The hierarchy contains the nodes but does not necessarily include additional state information to track which data is being browsed or cached by potentially multiple browsing machines.

[0022] Remote Machine 112 manages the hierarchy of nodes 122. Requests for nodes are received by remote machine 112 to fetch data from the hierarchy of nodes 122 and send the data to the requesting machine such as local machine 142. The remote machine 112 may be stateless with regard to the data being requested. That is, the remote machine 112 does not maintain a record of past requests. The request for data to the remote machine 112 is idempotent.

[0023] Display Screen 130 is a screen of a display device communicatively coupled to the local machine 142. Display area 132 is a display window in a region of the

display screen 130 in which an ordered list of data items of the hierarchical tree nodes are displayed and browsed. The ordered list of data items includes a number of data items according to the number of lines of data that can be displayed simultaneously in the display window.

[0024] Display Area 134 is a separate portion of the screen that may display meta data for the nodes displayed in Display Area 132. Examples of metadata that may be displayed in Display Area B include an ordered list of ancestor nodes in the hierarchy for a selected node. Other information that may be displayed in Display Area B include attributes of a selected node such as value and type, creation time, owner, etc. A search interface may be displayed in Display Area 134.

[0025] Local Machine 142 hosts a browser used by a user to browse the data in the hierarchy of nodes 122. A user views nodes in display area 132 and may navigate by scrolling to other nodes in an ordered list of nodes. As new nodes are scrolled to in the display window, the local machine may request that remote machine 112 send the to-be-displayed nodes to local machine 142 so that the nodes can be displayed in display area 132. A user may interact with display area 134 to specify a target node for viewing when the node is not within convenient scrolling distance from the nodes shown in display area 132. For example, display area 134 may contain a list of bookmarked nodes, recently viewed nodes, or a search bar in which the name of the target node may be specified. In response to the user requesting to view a particular target node, local machine 142 may determine which nodes are to be displayed next to the target node and may fetch those additional nodes from remote machine 112 as needed.

[0026] Cache of Partial Hierarchical Data 144 is a cache local to, and managed by, local machine 142. Local machine 142 stores copies of a subset of the nodes in the hierarchy of nodes 122 as the nodes are fetched from the remote machine. For every requested node in the local cache, the cache may include copies of ancestors, siblings, descendants, or other types of related nodes as specified in a policy. As an example, for every requested node in the local cache, the cache may include all ancestors of the requested node, but the cache may not necessarily include all the sibling nodes of the requested nodes. In another example, for every requested node in the local cache, the cache may include ancestor nodes up to *n* levels up and all the sibling nodes of the requested nodes. In addition to the data in the nodes that are copied in the cache, cached nodes may also include additional metadata that tracks the browsing state of the data in the browser. This metadata may not be known to the remote machine storing the master hierarchical dataset. For example, each internal node (that is, nodes having at least one child node) may have an associated expansion status indicator. The expansion status indicates whether child nodes of an internal node are to be displayed. In an embodiment, cache of partial hierarchical data 144 may also store a list of nodes for which the expansion status is expanded. This list of expanded nodes may be separate from the cached nodes, and the list may persist in the cache even when a node in the list may be evicted from the cache. In an embodiment, the expanded node list may include only those nodes that have been cached at least once during the browsing session and whose expansion status is expanded. Nodes that have not yet been cached during the browsing session and nodes having a collapsed status may not be in the list. Alternatively, the list of expanded nodes may be further restricted to only include nodes that have been evicted by the cache having an expanded status. In such an embodiment, a node may be added to the list of expanded nodes when an expanded node is evicted from the cache. When a node that has been previously evicted from the cache is re-fetched and entered into the cache, its expansion status may be set to expanded if the node appears in the list of nodes with expansion status. In an embodiment, a node may be removed from the list of expanded nodes when the node is stored in the cache, because the node's expansion state is stored within the cached node itself.

[0027] In an embodiment in which the nodes of the hierarchy are ordered according to the linearization rules described above, child nodes of a particular expanded internal node may appear in an ordered list of nodes before a sibling node of the particular internal node. However, if the particular node has an expansion status of "collapsed", the sibling node may appear immediately after the particular node without intervening child nodes.

[0028] Some of the nodes in the hierarchy of nodes 122 may not have corresponding copies in the cache. A placeholder may be used to represent a set of contiguous ordered nodes to indicate that the master hierarchy includes nodes that are not represented in the cache. When traversing the cached hierarchy for nodes to display in an ordered sequence, encountering a placeholder may cause the local machine to request copies of the missing nodes and to enter the fetched nodes into the cache.

[0029] Figure 1B illustrates Hierarchy of Nodes 122 in more detail, in accordance with one or more embodiments. The labels of the nodes in the hierarchy are selected, in the illustrated example, to facilitate identifying nodes. The illustrated example should not be construed to limit the scope of any of the claims. Each node has a unique identity when using the labels of its ancestors as a prefix. For example, the number in the node is an indication of the order of sibling nodes. Thus, a node with "1" is the first sibling node and a node with "2" is the second sibling node. The letter in the node indicates the level in the hierarchy. Thus, node 1B is a descendent of node 1A. Node 1A is the root of the hierarchy, and has 9999 child nodes in order f 1B, 2B, ..., 9999B}. The ellipses in the diagram indicate nodes that exist in the hierarchy, but representation has been left out of the diagram for lack of space. For example, node 1B has child nodes 4C, 5C, 6C etc., but child nodes 4C, 5C, 6C, etc. are not depicted in the figure. Nodes 1B, 2B, and 9999B each have 9999 child nodes as well. This hierarchy has

a very large number of nodes.

[0030] Figure 1C illustrates a local cache of nodes corresponding to a subset of the nodes in the hierarchical tree of data illustrated in Figure 1B, in accordance with one or more embodiments. Labeled nodes indicate copies of the corresponding nodes in the hierarchy of Figure 1B, and the thick bordered rectangles represent placeholders. The placeholder identifies a set of one or more sibling nodes that are contiguous in the ordered list of sibling nodes and that are not cached. For example, the entire subtree of 2B is not stored in the cache. Also missing from the cache are:

- child nodes of node 1B: {81C through 898C} and {1000C to 9999C} and
- child nodes of 10B: {1C to 19C} and {31C to 9999C}

[0031] Figure 2 is a flow diagram that illustrates retrieving and displaying a data item of a target node, in accordance with one or more embodiments. In Operation 210, an ordered sequence of nodes that is a portion of the hierarchy of nodes 122 is displayed in display area 132. The display window can present a certain number of lines of data items. In an embodiment, the local machine may pre-fetch some portion of the hierarchy for display in display area 132 at startup time, before Operation 210 is performed.

[0032] In Operation 220, the user requests to view a particular node in the hierarchy that is not shown in the display window, and thus, cannot be selected from display area 132. A user may choose the option of using display area 134 to select a node for display in display area 132 when the node of interest is not within reasonable scrolling distance of the nodes shown in the display window. In other words, the target node of interest may be located in a subtree that is far away in the hierarchy from the displayed nodes. A target node may be specified by interacting with display area 134 to select a bookmarked node, a recently viewed node, or entering a node name in a search bar to navigate directly to the target node without needing to scroll.

[0033] In Operation 230, the local machine searches for the target node in the local cache. If the target node is not stored in the local cache, then in Operation 240, the local machine may request a copy of the target node from remote machine 112 and store the copy of the target node in the cache. In addition, ancestors of the target node that are not already in the cache may be fetched and placed in the cache.

[0034] The hierarchy of cached nodes is used to determine the data items to display in display area 132 in Operation 250 in response to a user request. The data items to be displayed includes an ordered sequence of nodes. In an embodiment, the data item for the target node may be placed approximately in the center of the display window. The number of data items for the nodes to be displayed with the target node may be based on a display configuration. The display configuration may include, for example, the size of the display window, the resolution of the display window, the size of each illustration of a node, the amount of space between nodes. Some of the nodes to be displayed may be missing in the cache, and instead represented by placeholders in the cache. Identifying the data items to be displayed is discussed in further detail in the section below, titled "IDENTIFYING NODES TO BE DISPLAYED."

[0035] In operation 260, nodes that are determined to be needed for display, but which are missing from the cache, may be fetched from the remote machine 112. As nodes are received from the remote machine, the nodes may be added to the cache and placeholders removed or revised to indicate the smaller set of nodes that are missing.

[0036] In Operation 270, the nodes to be displayed, including the target node, are displayed in display area 132. For example, referring to Figure 1C, if node 899C is selected for display, information about nodes represented by the placeholder may be needed. For example, node 898C is missing from the cache. Node 898C would have to be fetched from the remote machine. Thus, when nodes from the ordered sequence of nodes to display are missing from the cache, Operations 230 through 260 may have to be repeated until the number of nodes in the ordered sequence fills the display window.

3. USER INTERFACE CONSIDERATIONS

[0037] The browser may support a user's ability to request a single operation to expand or collapse (a) an individual node or (b) all nodes.

[0038] The expansion state of a node entering the cache may be set to a default value. In an embodiment, the default expansion state is *not expanded* (also referred to herein as *collapsed*). A user may request expansion of a node by choosing (e.g. clicking on) the node in the display window. When a user chooses an individual node to be expanded, the node's expansion state indicator value may be set to *expanded,* and the child nodes of the selected node may be added into the displayed list of nodes. The expansion state indicator may remain as expanded until the user requests the node to be collapsed.

[0039] Selecting a node that is randomly accessed through a search interface or other means that do not include choosing from within the display window may cause the value of the expansion state indicator for all ancestors of the selected node to be set to an expanded state. The displayed list of nodes appear to the user as though the user manually expanded all of the ancestor nodes and scrolled to the selected node.

[0040] In an embodiment, a user may request to expand all, which changes the expansion state of all nodes in the hierarchy to *expanded.* Upon expanding all nodes in the hierarchy, the browsing state may be changed for all nodes, not only the nodes that are being displayed or cached at the time the request is made. In an embodiment, a global expansion state indicator may indicate

whether all nodes are to be displayed as expanded. If the global expansion state indicator indicates expansion mode, every node is displayed as though its state is expanded regardless of the value of the node's individual expansion state identifier value.

[0041] In an embodiment, a user may request to expand down, which may change the expansion state of all nodes in the subtree rooted at the chosen node to be *expanded.* In an embodiment, a subtree expansion indicator may be stored in the cache that identifies the node that is the root of the subtree to be expanded. To determine whether a node to be displayed should be expanded, the system may consult the subtree expansion indicators to determine whether the node is in the subtree of any of the nodes identified by the subtree expansion indicators.

[0042] Similarly, the user may request to collapse all, which may cause all nodes to become collapsed. The expansion state indicator may be set to collapsed, and a list of individual expanded nodes may be emptied. In an embodiment, subtree expansion indicators may be deleted.

4. IDENTIFYING NODES TO BE DISPLAYED

[0043] According to the invention, the system identifies the nodes to be displayed based on a requested node. The nodes to be displayed represent a particular section of the hierarchical tree of nodes that is to be displayed. A section of the hierarchical tree of nodes, as referred to herein, may include any contiguous or non-contiguous set of nodes in the hierarchical tree of nodes.

[0044] A requested node is generally displayed with nodes that are directly or indirectly related to the requested node. As an example, in response to a request for a particular node, the particular node may be illustrated in the center of a display. The system may be configured, by default to display up to five ancestors of the requested node, two siblings of the requested node, and no descendants. The five ancestor nodes may be determined by traversing up to five levels from the requested node in the hierarchical tree of nodes. The two siblings may be determined based on the position of the requested node in an ordered sequence of siblings. The two nodes following the requested node in an ordered sequence of siblings may be identified as the siblings to displayed. Displaying no descendants initially means that the system is configured to initially display requested nodes in a collapsed state. An alternate configuration may display requested nodes in an expanded state. As an example, the first x descendants in an ordered sequence of descendants of the requested node may be included with an initial display of the requested node.

[0045] In an embodiment, the system identifies the nodes to be displayed when a particular displayed node is expanded. In response to the request to expand a particular node, the system determines the number of descendant nodes that are to be displayed. As an example, the system may be configured to initially display the first four descendant nodes without displaying other descendant nodes of the expanded node. A display of the other descendant nodes may be triggered by a user scrolling down the list of descendant nodes.

[0046] In an embodiment, each node of the hierarchical tree of nodes may be displayed on a separate horizontal line of the display. A user may scroll down to nodes earlier in the ordered list of nodes not previously displayed or may scroll up to see nodes later in the ordered list that were not previously displayed. Responsive to detecting the scrolling, the system may identify one or more nodes to be added to the display as well as one or more nodes to remove from the display because the nodes no longer fit on the display.

[0047] The nodes to be displayed, as explained above, are limited to a specific number of nodes based on a configuration associated with a display window. As an example, the number of nodes that can be displayed in a window may be a function of a size and/or resolution of the window, a size of an icon representing each node within the window, a font size of a node identifier, and a spacing between nodes. The system may determine, for example, that ten nodes can be displayed with a requested node. Based on a priority and/or policy, the system may select ten nodes, of the nodes related to the requested node, as the to-be-displayed nodes.

[0048] In an embodiment, the approach described herein efficiently creates an ordered list of nodes to fill a display window having a particular node in the ordered list considering the expansion status of internal nodes in the hierarchy. The particular node may be placed in the center of the display window, though any position within the display window may be selected. A basic approach to creating the ordered list of nodes to display, which is not covered by the present claims, would be to express the entire hierarchy as an ordered list based on the expansion status of each node. The entire hierarchy may be searched in a pre-defined order, treating unexpanded internal nodes as though the internal nodes were leaf nodes. In other words, when a node that is a root of a subtree is not expanded, the subtree is not searched (and child nodes of the subtree root are not included in the list). The particular node may be located in the list ordered list, and a number of nodes before the particular node in the list, and a number of nodes after the particular node in the list, may be selected for display. When the particular node is to appear in the center of the display window, the number of nodes before and after the particular node may be roughly half of the maximum number of lines in the display window. However, in a hierarchy of millions of nodes, creating an ordered list for the entire hierarchy may be prohibitively time consuming, especially if the user has chosen to "expand all" nodes.

[0049] A more efficient approach to create the ordered list, which is used according to the presently claimed inventon, is to first search the subtrees closest to the particular node in the hierarchy, and to expand the search

to further relatives only as needed to fill the display list. In an embodiment, the hierarchy may be searched in an order corresponding to the desired order of data items to display starting with the immediate parent node of the particular node. As explained earlier, the parent node may be expanded in this embodiment because the particular node is to be displayed. If, in this embodiment, there are enough sibling nodes of the particular node to fill the ordered list, then no further search is needed. If, in this embodiment, the number of nodes in the ordered list created by exploring the immediate parent node does not fill the display list, then the search may be performed on the grandparent node, and so on until the ordered list includes a sufficient number of nodes to fill the display window.

[0050] In an embodiment, two separate ordered lists may be created: one list to display before the particular node and one list to display after the particular node. Each list may be formed by searching a different subtree. For example, if the particular node is the last of a set of ordered sibling nodes, then the list of nodes to display after the particular node may come from searching a different (aunt/uncle) subtree rather than from exploring the particular node's parent node.

5. WORKING WITH PLACEHOLDERS

[0051] According to the invention, creating the ordered list as described above is performed using the portions of the hierarchy in the local cache. While searching and ordering nodes of a subtree, a placeholder may be encountered that indicates a contiguous set of child nodes that exist but are not in the cache. To complete the search, at least a portion of the nodes represented by the placeholder may be fetched from the remote machine, added to the cache, and the placeholder may be removed or altered before the search continues. In an embodiment, all the nodes represented by a placeholder may be fetched, and the placeholder may be removed entirely. In an embodiment, a portion of the nodes represented by the placeholder may be fetched in the which the number of fetched nodes does not exceed: (a) a prefetching limit or (b) the number of nodes in the ordered list that can be displayed. The placeholder may be updated to represent the nodes that have not yet been fetched.

[0052] In an embodiment, if a user selects a node to display by entering an identifier of the selected node in a search interface, the selected node may reside in a subtree that has not yet been cached and/or expanded. Thus, when a selected node is randomly accessed, each of the ancestor nodes' expansion status may be set to expanded. Child nodes that were recently expanded, but not yet represented in the cache, may be fetched and cached or may be represented by a placeholder until those child nodes are determined to be needed for display.

[0053] When the cache runs out of space, nodes in the cache may be selected for eviction according to a cache management policy, and the node selected for eviction may be replaced by a placeholder in the cached hierarchy.

6. FETCHING NODES FROM THE REMOTE MACHINE

[0054] When at least one node needs to be retrieved from the remote machine, the number of nodes to fetch may depend on several factors. The minimal number of nodes to retrieve is the number of nodes that are needed to fill the display list. This minimal number of nodes are the nodes to be displayed as explained above, in section 4, titled "identifying nodes to be displayed." If a group of contiguous nodes are missing, all of the nodes in the group of contiguous nodes may be fetched in the same request. However, if a group of nodes are needed to fill the display list, and the nodes are not contiguous in the ordered list, the local machine may request a number of contiguous nodes that includes at least one node that needs to be fetched. The number of contiguous nodes to be fetched may be determined as a function of the number of nodes that can be simultaneously displayed in the display window. For display purposes, the number of nodes needed in the ordered sequence does not exceed the number nodes that can be simultaneously displayed the display window. Fetching contiguous nodes, some already cached, may be more efficient than sending individual requests for non-contiguous placeholders within a common subtree.

[0055] In addition, the number of nodes to be fetched may be determined based on performance considerations. To reduce the number of fetch requests to the remote machine, more nodes than needed may be sent in response to each fetch request. Although the nodes to be displayed may be a contiguous set of nodes within the ordered list of nodes of the hierarchy, additional nodes, that are next to the contiguous set of nodes to display, may be prefetched. If the list of nodes in the resulting display window is scrolled, then the pre-fetched nodes may already be in the cache. Pre-fetching may be performed when retrieving the additional nodes would not have an adverse impact on performance. That number may be determined experimentally and configured into the system. Alternatively, that number may be based on a current load, and may change dynamically over time.

7. COMPUTER NETWORKS AND CLOUD NETWORKS

[0056] In one or more embodiments, a computer network provides connectivity among a set of nodes. The nodes may be local to and/or remote from each other. The nodes are connected by a set of links. Examples of links include a coaxial cable, an unshielded twisted cable, a copper cable, an optical fiber, and a virtual link.

[0057] A subset of nodes implements the computer network. Examples of such nodes include a switch, a

router, a firewall, and a network address translator (NAT). Another subset of nodes uses the computer network. Such nodes (also referred to as "hosts") may execute a client process and/or a server process. A client process makes a request for a computing service (such as, execution of a particular application, and/or storage of a particular amount of data). A server process responds by executing the requested service and/or returning corresponding data.

[0058] A computer network may be a physical network, including physical nodes connected by physical links. A physical node is any digital device. A physical node may be a function-specific hardware device, such as a hardware switch, a hardware router, a hardware firewall, and a hardware NAT. Additionally or alternatively, a physical node may be a generic machine that is configured to execute various virtual machines and/or applications performing respective functions. A physical link is a physical medium connecting two or more physical nodes. Examples of links include a coaxial cable, an unshielded twisted cable, a copper cable, and an optical fiber.

[0059] A computer network may be an overlay network. An overlay network is a logical network implemented on top of another network (such as, a physical network). Each node in an overlay network corresponds to a respective node in the underlying network. Hence, each node in an overlay network is associated with both an overlay address (to address to the overlay node) and an underlay address (to address the underlay node that implements the overlay node). An overlay node may be a digital device and/or a software process (such as, a virtual machine, an application instance, or a thread) A link that connects overlay nodes is implemented as a tunnel through the underlying network. The overlay nodes at either end of the tunnel treat the underlying multi-hop path between them as a single logical link. Tunneling is performed through encapsulation and decapsulation.

[0060] In an embodiment, a client may be local to and/or remote from a computer network. The client may access the computer network over other computer networks, such as a private network or the Internet. The client may communicate requests to the computer network using a communications protocol, such as Hypertext Transfer Protocol (HTTP). The requests are communicated through an interface, such as a client interface (such as a web browser), a program interface, or an application programming interface (API).

[0061] In an embodiment, a computer network provides connectivity between clients and network resources. Network resources include hardware and/or software configured to execute server processes. Examples of network resources include a processor, a data storage, a virtual machine, a container, and/or a software application. Network resources are shared amongst multiple clients. Clients request computing services from a computer network independently of each other. Network resources are dynamically assigned to the requests and/or clients on an on-demand basis. Network resources assigned to each request and/or client may be scaled up or down based on, for example, (a) the computing services requested by a particular client, (b) the aggregated computing services requested by a particular tenant, and/or (c) the aggregated computing services requested of the computer network. Such a computer network may be referred to as a "cloud network."

[0062] In an embodiment, a service provider provides a cloud network to one or more end users. Various service models may be implemented by the cloud network, including but not limited to Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS). In SaaS, a service provider provides end users the capability to use the service provider's applications, which are executing on the network resources. In PaaS, the service provider provides end users the capability to deploy custom applications onto the network resources. The custom applications may be created using programming languages, libraries, services, and tools supported by the service provider. In IaaS, the service provider provides end users the capability to provision processing, storage, networks, and other fundamental computing resources provided by the network resources. Any arbitrary applications, including an operating system, may be deployed on the network resources.

[0063] In an embodiment, various deployment models may be implemented by a computer network, including but not limited to a private cloud, a public cloud, and a hybrid cloud. In a private cloud, network resources are provisioned for exclusive use by a particular group of one or more entities (the term "entity" as used herein refers to a corporation, organization, person, or other entity). The network resources may be local to and/or remote from the premises of the particular group of entities. In a public cloud, cloud resources are provisioned for multiple entities that are independent from each other (also referred to as "tenants" or "customers"). The computer network and the network resources thereof are accessed by clients corresponding to different tenants. Such a computer network may be referred to as a "multi-tenant computer network." Several tenants may use a same particular network resource at different times and/or at the same time. The network resources may be local to and/or remote from the premises of the tenants. In a hybrid cloud, a computer network comprises a private cloud and a public cloud. An interface between the private cloud and the public cloud allows for data and application portability. Data stored at the private cloud and data stored at the public cloud may be exchanged through the interface. Applications implemented at the private cloud and applications implemented at the public cloud may have dependencies on each other. A call from an application at the private cloud to an application at the public cloud (and vice versa) may be executed through the interface.

[0064] In an embodiment, tenants of a multi-tenant computer network are independent of each other. For example, a business or operation of one tenant may be separate from a business or operation of another tenant.

Different tenants may demand different network requirements for the computer network. Examples of network requirements include processing speed, amount of data storage, security requirements, performance requirements, throughput requirements, latency requirements, resiliency requirements, Quality of Service (QoS) requirements, tenant isolation, and/or consistency. The same computer network may need to implement different network requirements demanded by different tenants.

[0065] In one or more embodiments, in a multi-tenant computer network, tenant isolation is implemented to ensure that the applications and/or data of different tenants are not shared with each other. Various tenant isolation approaches may be used.

[0066] In an embodiment, each tenant is associated with a tenant ID. Each network resource of the multi-tenant computer network is tagged with a tenant ID. A tenant is permitted access to a particular network resource only if the tenant and the particular network resources are associated with a same tenant ID.

[0067] In an embodiment, each tenant is associated with a tenant ID. Each application, implemented by the computer network, is tagged with a tenant ID. Additionally or alternatively, each data structure and/or dataset, stored by the computer network, is tagged with a tenant ID. A tenant is permitted access to a particular application, data structure, and/or dataset only if the tenant and the particular application, data structure, and/or dataset are associated with a same tenant ID.

[0068] As an example, each database implemented by a multi-tenant computer network may be tagged with a tenant ID. Only a tenant associated with the corresponding tenant ID may access data of a particular database. As another example, each entry in a database implemented by a multi-tenant computer network may be tagged with a tenant ID. Only a tenant associated with the corresponding tenant ID may access data of a particular entry. However, the database may be shared by multiple tenants.

[0069] In an embodiment, a subscription list indicates which tenants have authorization to access which applications. For each application, a list of tenant IDs of tenants authorized to access the application is stored. A tenant is permitted access to a particular application only if the tenant ID of the tenant is included in the subscription list corresponding to the particular application.

[0070] In an embodiment, network resources (such as digital devices, virtual machines, application instances, and threads) corresponding to different tenants are isolated to tenant-specific overlay networks maintained by the multi-tenant computer network. As an example, packets from any source device in a tenant overlay network may only be transmitted to other devices within the same tenant overlay network. Encapsulation tunnels are used to prohibit any transmissions from a source device on a tenant overlay network to devices in other tenant overlay networks. Specifically, the packets, received from the source device, are encapsulated within an outer packet. The outer packet is transmitted from a first encapsulation tunnel endpoint (in communication with the source device in the tenant overlay network) to a second encapsulation tunnel endpoint (in communication with the destination device in the tenant overlay network). The second encapsulation tunnel endpoint decapsulates the outer packet to obtain the original packet transmitted by the source device. The original packet is transmitted from the second encapsulation tunnel endpoint to the destination device in the same particular overlay network.

8. HARDWARE OVERVIEW

[0071] According to one embodiment, the techniques described herein are implemented by one or more special-purpose computing devices. The special-purpose computing devices may be hard-wired to perform the techniques, or may include digital electronic devices such as one or more application-specific integrated circuits (ASICs), field programmable gate arrays (FPGAs), or network processing units (NPUs) that are persistently programmed to perform the techniques, or may include one or more general purpose hardware processors programmed to perform the techniques pursuant to program instructions in firmware, memory, other storage, or a combination. Such special-purpose computing devices may also combine custom hard-wired logic, ASICs, FPGAs, or NPUs with custom programming to accomplish the techniques. The special-purpose computing devices may be desktop computer systems, portable computer systems, handheld devices, networking devices or any other device that incorporates hard-wired and/or program logic to implement the techniques.

[0072] For example, Figure 3 is a block diagram that illustrates a computer system 300 upon which an embodiment of the invention may be implemented. Computer system 300 includes a bus 302 or other communication mechanism for communicating information, and a hardware processor 304 coupled with bus 302 for processing information. Hardware processor 304 may be, for example, a general purpose microprocessor.

[0073] Computer system 300 also includes a main memory 306, such as a random access memory (RAM) or other dynamic storage device, coupled to bus 302 for storing information and instructions to be executed by processor 304. Main memory 306 also may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor 304. Such instructions, when stored in non-transitory storage media accessible to processor 304, render computer system 300 into a special-purpose machine that is customized to perform the operations specified in the instructions.

[0074] Computer system 300 further includes a read only memory (ROM) 308 or other static storage device coupled to bus 302 for storing static information and instructions for processor 304. A storage device 310, such as a magnetic disk or optical disk, is provided and coupled

to bus 302 for storing information and instructions.

[0075] Computer system 300 may be coupled via bus 302 to a display 312, such as a cathode ray tube (CRT), for displaying information to a computer user. An input device 314, including alphanumeric and other keys, is coupled to bus 302 for communicating information and command selections to processor 304. Another type of user input device is cursor control 316, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor 304 and for controlling cursor movement on display 312. This input device typically has two degrees of freedom in two axes, a first axis (e.g., x) and a second axis (e.g., y), that allows the device to specify positions in a plane.

[0076] Computer system 300 may implement the techniques described herein using customized hard-wired logic, one or more ASICs or FPGAs, firmware and/or program logic which in combination with the computer system causes or programs computer system 300 to be a special-purpose machine. According to one embodiment, the techniques herein are performed by computer system 300 in response to processor 304 executing one or more sequences of one or more instructions contained in main memory 306. Such instructions may be read into main memory 306 from another storage medium, such as storage device 310. Execution of the sequences of instructions contained in main memory 306 causes processor 304 to perform the process steps described herein. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions.

[0077] The term "storage media" as used herein refers to any non-transitory media that store data and/or instructions that cause a machine to operate in a specific fashion. Such storage media may comprise non-volatile media and/or volatile media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device 310. Volatile media includes dynamic memory, such as main memory 306. Common forms of storage media include, for example, a floppy disk, a flexible disk, hard disk, solid state drive, magnetic tape, or any other magnetic data storage medium, a CD-ROM, any other optical data storage medium, any physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, NVRAM, any other memory chip or cartridge, content-addressable memory (CAM), and ternary content-addressable memory (TCAM).

[0078] Storage media is distinct from but may be used in conjunction with transmission media. Transmission media participates in transferring information between storage media. For example, transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus 302. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infra-red data communications.

[0079] Various forms of media may be involved in carrying one or more sequences of one or more instructions to processor 304 for execution. For example, the instructions may initially be carried on a magnetic disk or solid state drive of a remote computer. The remote computer can load the instructions into its dynamic memory and send the instructions over a telephone line using a modem. A modem local to computer system 300 can receive the data on the telephone line and use an infra-red transmitter to convert the data to an infra-red signal. An infra-red detector can receive the data carried in the infra-red signal and appropriate circuitry can place the data on bus 302. Bus 302 carries the data to main memory 306, from which processor 304 retrieves and executes the instructions. The instructions received by main memory 306 may optionally be stored on storage device 310 either before or after execution by processor 304.

[0080] Computer system 300 also includes a communication interface 318 coupled to bus 302. Communication interface 318 provides a two-way data communication coupling to a network link 320 that is connected to a local network 322. For example, communication interface 318 may be an integrated services digital network (ISDN) card, cable modem, satellite modem, or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, communication interface 318 may be a local area network (LAN) card to provide a data communication connection to a compatible LAN. Wireless links may also be implemented. In any such implementation, communication interface 318 sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

[0081] Network link 320 typically provides data communication through one or more networks to other data devices. For example, network link 320 may provide a connection through local network 322 to a host computer 324 or to data equipment operated by an Internet Service Provider (ISP) 326. ISP 326 in turn provides data communication services through the world wide packet data communication network now commonly referred to as the "Internet" 328. Local network 322 and Internet 328 both use electrical, electromagnetic or optical signals that carry digital data streams. The signals through the various networks and the signals on network link 320 and through communication interface 318, which carry the digital data to and from computer system 300, are example forms of transmission media.

[0082] Computer system 300 can send messages and receive data, including program code, through the network(s), network link 320 and communication interface 318. In the Internet example, a server 330 might transmit a requested code for an application program through Internet 328, ISP 326, local network 322 and communication interface 318.

[0083] The received code may be executed by processor 304 as it is received, and/or stored in storage device 310, or other non-volatile storage for later execution.

[0084] Embodiments are directed to a system with one

or more devices that include a hardware processor and that are configured to perform any of the operations described herein and/or recited in any of the claims below.

[0085] In an embodiment, a non-transitory computer readable storage medium comprises instructions which, when executed by one or more hardware processors, causes performance of any of the operations described herein and/or recited in any of the claims.

Claims

1. One or more non-transitory computer readable media storing instructions which, when executed by one or more hardware processors (304), cause:

   displaying, in a display window (132A, 132B), a first subset of nodes which represents a first section of a hierarchical tree of nodes (122), the hierarchical tree of nodes (122) being stored on a remote machine (112);
   receiving a request for displaying a target node of the hierarchical tree of nodes (122), the target node not being displayed when the request is received; based on a configuration corresponding to the display window (132A, 132B),
   determining an ordered list of nodes to fill the display window and which represents a second section of the hierarchical tree (122) which is to be displayed in the display window (132A, 132B) when displaying the target node, the ordered list of nodes comprising the target node and a second subset of nodes in the hierarchical tree of nodes (122), wherein said determining said ordered list of nodes further involves using a local cache storing portions of the hierarchical tree of nodes as locally cached hierarchy of nodes, wherein a placeholder is used in the locally cached hierarchy of nodes to represent a set of contiguous ordered nodes of the hierarchical tree of nodes stored on the remote machine that are not stored in the local cache;
   wherein said determining said ordered list of nodes using the local cache involves searching subtrees in the locally cached hierarchical tree of nodes (112) closest to the target node, and successively expanding the search to more distant relatives of the target node only as needed until a sufficient number of nodes to fill the display window (132A, 132B) has been determined;
   wherein, while searching the locally cached hierarchical tree of nodes, a placeholder is encountered that indicates a contiguous set of child nodes that exist in the hierarchy of nodes stored at the remote machine but that are not locally cached, at least a portion of the child nodes is fetched from the remote machine;
   concurrently displaying the target node and the second subset of nodes.

2. The media of claim 1, wherein said determining said ordered list of nodes starts with the immediate parent node of the target node to determine sibling nodes of the target node and proceeds, if needed, to the grandparent node of the target node and to zero or more further ancestor nodes, until said sufficient number of nodes to fill the display window (132A, 132B) has been determined.

3. The media of claim 1, wherein said ordered list of nodes includes one or more ancestor nodes of the target node, and wherein said one or more ancestor nodes of the target node are determined by traversing upwards from the target node in the locally cached hierarchical tree of nodes (122).

4. The media of any of claims 1-3, wherein said sufficient number of nodes to fill the display window (132A, 132B) is determined based on the configuration corresponding to the display window (132A, 132B), and wherein the configuration corresponding to the display window (132A, 132B) comprises one or more of: a size of the display window, a resolution of the display window (132A, 132B), a size of node illustrations, and an amount of space between node illustrations.

5. The media of any of claims 1-4 wherein caching a particular node of the one or more nodes fetched from the remote machine (112) further comprises:
   setting the value of an expansion state indicator for the particular node to a default value, wherein:

   a value that indicates that the state of the particular node is expanded causes child nodes of the particular node to be displayed contiguous to the particular node; and
   a value that indicates that the state of the particular node is collapsed causes the particular node to be displayed without displaying child nodes of the particular node.

6. The media of claim 5, wherein the instructions further cause:

   receiving a request to expand all nodes in the hierarchical tree of nodes (122);
   responsive to receiving the request to expand all nodes in the hierarchical tree of nodes (122), setting a value of a global expansion state indicator to indicate that all nodes representing nodes in the hierarchical tree of nodes (122) are to be displayed as expanded;
   determining when the request is received that a displayed node is being displayed and child

nodes of the displayed node are not being displayed;
fetching one or more child nodes of the displayed node that are missing from the cache without fetching child nodes of other nodes that are not being displayed;
changing the displayed list of nodes to include a portion of the child nodes of the particular node; and
adding the fetched one or more child nodes to the hierarchy of cached nodes (144).

7. The media of claim 5, wherein the remote machine (112) responds to requests for nodes in the hierarchical tree of nodes (122) without tracking:

(a) which nodes have been sent to the requester or
(b) state of the hierarchy of cached nodes (144) maintained by the requester

8. The media of any of claims 1-7, wherein:

each node in the hierarchical tree of nodes (122) has an ordered sequence of zero or more child nodes, wherein one or more child nodes of the ordered sequence of child nodes are displayed in order of their sequence;
child nodes of unexpanded nodes are omitted from the ordered list of nodes; and
determining the ordered list of nodes further comprises selecting contiguous nodes in the ordered list of nodes, wherein:

the selected contiguous nodes includes the target node; and
the number of contiguous nodes is based on the number of nodes that can be displayed together.

9. The media of any of claims 1-8, wherein the fetching further comprises fetching a further subset of nodes that are adjacent to the second subset of nodes in the hierarchical tree of nodes (122), wherein the further subset of nodes is not displayed with the target node.

10. A method comprising the operations as recited in any of claims 1-9.

11. A system comprising:

at least one hardware device including a processor (304); and
the system configured to perform the operations as recited in any of claims 1-9.

Patentansprüche

1. Ein oder mehrere nicht-transitorische computerlesbare Medien, die Anweisungen speichern, die, wenn sie von einem oder mehreren Hardware-Prozessoren (304) ausgeführt werden, bewirken:

Anzeigen, in einem Anzeigefenster (132A, 132B), einer ersten Untergruppe von Knoten, die einen ersten Abschnitt eines hierarchischen Baums von Knoten (122) darstellt, wobei der hierarchische Baum von Knoten (122) auf einem entfernten Rechner (112) gespeichert ist;
Empfangen einer Anforderung zum Anzeigen eines Zielknotens des hierarchischen Baums von Knoten (122), wobei der Zielknoten nicht angezeigt wird, wenn die Anforderung empfangen wird;
Bestimmen, und zwar beruhend auf einer dem Anzeigefenster (132A, 132B) entsprechenden Konfiguration, einer geordneten Liste von Knoten zum Füllen des Anzeigefensters, die einen zweiten Abschnitt des hierarchischen Baums (122) darstellt, der in dem Anzeigefenster (132A, 132B) angezeigt werden soll, wenn der Zielknoten angezeigt wird, wobei die geordnete Liste von Knoten den Zielknoten und eine zweite Untergruppe von Knoten in dem hierarchischen Baum von Knoten (122) aufweist,
wobei das Bestimmen der geordneten Liste von Knoten ferner umfasst, dass ein lokaler Pufferspeicher verwendet wird, der Teile des hierarchischen Baums von Knoten als eine lokal gepufferte Hierarchie von Knoten speichert, wobei ein Platzhalter in der lokal gepufferten Hierarchie von Knoten verwendet wird, um einen Satz von benachbarten geordneten Knoten des auf dem entfernten Rechner gespeicherten hierarchischen Baums von Knoten darzustellen, die nicht in dem lokalen Pufferspeicher gespeichert sind;
wobei das Bestimmen der geordneten Liste von Knoten unter Verwendung des lokalen Pufferspeichers umfasst, dass Teilbäume in dem lokal gepufferten hierarchischen Baum von Knoten (112) durchsucht werden, die dem Zielknoten am nächsten sind, und dass die Suche sukzessive auf weiter entfernte Verwandte des Zielknotens nur nach Bedarf erweitert wird, bis eine ausreichende Anzahl von Knoten zum Füllen des Anzeigefensters (132A, 132B) bestimmt worden ist;
wobei beim Durchsuchen des lokal gepufferten hierarchischen Baums von Knoten ein Platzhalter angetroffen wird, der einen Satz von benachbarten Kindknoten angibt, die in der auf dem entfernten Rechner gespeicherten Hierarchie von Knoten existieren, aber nicht lokal gepuffert

sind, wobei mindestens ein Teil der Kindknoten von dem entfernten Rechner geholt wird; und
gleichzeitiges Anzeigen des Zielknotens und der zweiten Untergruppe von Knoten.

2. Ein oder mehrere Medien nach Anspruch 1, wobei das Bestimmen der geordneten Liste von Knoten mit dem unmittelbaren Elternknoten des Zielknotens beginnt, um Geschwisterknoten des Zielknotens zu bestimmen, und bei Bedarf mit dem Großelternknoten des Zielknotens und mit null oder mehr weiteren Vorfahrenknoten fortfährt, bis die ausreichende Anzahl von Knoten zum Füllen des Anzeigefensters (132A, 132B) bestimmt worden ist.

3. Ein oder mehrere Medien nach Anspruch 1, wobei die geordnete Liste von Knoten einen oder mehrere Vorfahrenknoten des Zielknotens aufweist, und wobei der eine oder die mehreren Vorfahrenknoten des Zielknotens bestimmt werden, indem in dem lokal gepufferten hierarchischen Baum von Knoten (122) ein Durchlauf vom Zielknoten nach oben durchgeführt wird.

4. Ein oder mehrere Medien nach einem der Ansprüche 1-3, wobei die ausreichende Anzahl von Knoten zum Füllen des Anzeigefensters (132A, 132B) beruhend auf der dem Anzeigefenster (132A, 132B) entsprechenden Konfiguration bestimmt wird, und wobei die dem Anzeigefenster (132A, 132B) entsprechende Konfiguration eines oder mehrere der folgenden Merkmale aufweist: eine Größe des Anzeigefensters, eine Auflösung des Anzeigefensters (132A, 132B), eine Größe von Knotenabbildungen, und ein Maß von Platz zwischen Knotenabbildungen.

5. Ein oder mehrere Medien nach einem der Ansprüche 1-4, wobei das Puffern eines bestimmten Knotens des einen oder der mehreren Knoten, die von dem entfernten Rechner (112) abgerufen wurden, ferner umfasst:
Setzen des Wertes eines Expansionszustandsindikators für den bestimmten Knoten auf einen Standardwert, wobei:

ein Wert, der angibt, dass der Zustand des bestimmten Knotens erweitert ist, bewirkt, dass Kindknoten des bestimmten Knotens benachbart dem bestimmten Knoten angezeigt werden; und
ein Wert, der angibt, dass der Zustand des bestimmten Knotens kollabiert ist, bewirkt, dass der bestimmte Knoten angezeigt wird, ohne dass Kindknoten des bestimmten Knotens angezeigt werden.

6. Ein oder mehrere Medien nach Anspruch 5, wobei die Anweisungen ferner bewirken:

Empfangen einer Anforderung, alle Knoten in dem hierarchischen Baum von Knoten (122) zu expandieren;
ansprechend auf den Empfang der Anforderung, alle Knoten in dem hierarchischen Baum von Knoten (122) zu expandieren, wird ein Wert eines globalen Expansionszustandsindikators gesetzt, um anzugeben, dass alle Knoten, die Knoten in dem hierarchischen Baum von Knoten (122) darstellen, expandiert angezeigt werden sollen;
Bestimmen, wenn die Anforderung empfangen wird, dass ein angezeigter Knoten angezeigt wird und Kindknoten des angezeigten Knotens nicht angezeigt werden;
Abrufen eines oder mehrerer Kindknoten des angezeigten Knotens, die im Pufferspeicher fehlen, ohne dass Kindknoten anderer Knoten, die nicht angezeigt werden, abgerufen werden;
Ändern der angezeigten Liste von Knoten, so dass ein Teil der Kindknoten des bestimmten Knotens aufgenommen wird; und
Hinzufügen des abgerufenen einen Kindknotens oder der abgerufenen mehreren Kindknoten zu der Hierarchie der gepufferten Knoten (144).

7. Ein oder mehrere Medien nach Anspruch 5, wobei der entfernte Rechner (112) auf Anforderungen für Knoten in dem hierarchischen Baum von Knoten (122) antwortet, ohne zu verfolgen:

(a) welche Knoten an den Anforderer gesendet wurden oder
(b) einen Zustand der von dem Anforderer unterhaltenen Hierarchie von gepufferten Knoten (144).

8. Ein oder mehrere Medien nach einem der Ansprüche 1-7, wobei:

jeder Knoten in dem hierarchischen Baum von Knoten (122) eine geordnete Folge von null oder mehr Kindknoten aufweist, wobei ein oder mehrere Kindknoten der geordneten Folge von Kindknoten in der Reihenfolge ihrer Abfolge angezeigt werden;
Kindknoten von nicht-expandierten Knoten in der geordneten Liste von Knoten weggelassen werden; und
das Bestimmen der geordneten Liste von Knoten ferner umfasst, dass benachbarte Knoten in der geordneten Liste von Knoten ausgewählt werden, wobei:

die ausgewählten benachbarten Knoten den Zielknoten aufweisen; und
die Anzahl der benachbarten Knoten auf

der Anzahl von Knoten beruht, die zusammen angezeigt werden können.

9. Ein oder mehrere Medien nach einem der Ansprüche 1-8, wobei das Abrufen ferner umfasst, dass eine weitere Untergruppe von Knoten, die der zweiten Untergruppe von Knoten in dem hierarchischen Baum von Knoten (122) benachbart sind, abgerufen wird, wobei die weitere Untergruppe von Knoten nicht mit dem Zielknoten angezeigt wird.

10. Verfahren, das die in einem der Ansprüche 1-9 angegebenen Operationen umfasst.

11. System, das aufweist:

mindestens eine Hardwareeinrichtung, die einen Prozessor (304) aufweist; und
wobei das System konfiguriert ist, um die in einem der Ansprüche 1-9 genannten Operationen auszuführen.

Revendications

1. Un ou plusieurs supports lisibles par ordinateur non transitoires conservant en mémoire des instructions qui, lorsqu'elles sont exécutées par un ou plusieurs processeurs matériels (304), amènent :

l'affichage, dans une fenêtre d'affichage (132A, 132B), d'un premier sous-ensemble de noeuds qui représente une première section d'un arbre hiérarchique de noeuds (122), l'arbre hiérarchique de noeuds (122) étant conservé en mémoire sur un ordinateur distant (112),
la réception d'une demande destinée à l'affichage d'un noeud cible de l'arbre hiérarchique de noeuds (122), le noeud cible n'étant pas affiché lorsque la demande est reçue,
en fonction d'une configuration correspondant à la fenêtre d'affichage (132A, 132B), la détermination d'une liste ordonnée de noeuds destinée à remplir la fenêtre d'affichage et qui représente une deuxième section de l'arbre hiérarchique (122) qui est destinée à être affichée dans la fenêtre d'affichage (132A, 132B) lors de l'affichage du noeud cible, la liste ordonnée de noeuds comprenant le noeud cible et un deuxième sous-ensemble de noeuds dans l'arbre hiérarchique de noeuds (122),
où ladite détermination de ladite liste ordonnée de noeuds implique en outre l'utilisation d'une mémoire tampon locale conservant en mémoire des parties de l'arbre hiérarchique de noeuds sous la forme d'une hiérarchie de noeuds placée dans une mémoire tampon locale, où un emplacement réservé est utilisé dans la hiérarchie de noeuds placée dans une mémoire tampon locale de façon à représenter un ensemble de noeuds ordonnés contigus de l'arbre hiérarchique de noeuds conservés en mémoire sur l'ordinateur distant qui ne sont pas conservés en mémoire dans la mémoire tampon locale,
où ladite détermination de ladite liste ordonnée de noeuds au moyen de la mémoire tampon locale implique la recherche de sous-arbres dans l'arbre hiérarchique de noeuds placé dans une mémoire tampon locale (112) les plus proches du noeud cible, et l'expansion successive de la recherche vers des parents plus distants du noeud cible uniquement en fonction des besoins jusqu'à ce qu'un nombre suffisant de noeuds destinés à remplir la fenêtre d'affichage (132A, 132B) a été déterminé,
où, pendant la recherche de l'arbre hiérarchique de noeuds placé dans une mémoire tampon locale, un emplacement réservé est rencontré qui indique un ensemble contigu de noeuds enfants qui existent dans la hiérarchie de noeuds conservée en mémoire au niveau de l'ordinateur distant mais qui ne sont pas placés dans une mémoire tampon locale, au moins une partie des noeuds enfants est extraite de l'ordinateur distant, et
l'affichage simultané du noeud cible et du deuxième sous-ensemble de noeuds.

2. Le support selon la Revendication 1, où ladite détermination de ladite liste ordonnée de noeuds débute avec le noeud parent immédiat du noeud cible de façon à déterminer des noeuds frères du noeud cible et se poursuit, si nécessaire, vers le noeud grand-parent du noeud cible et vers zéro ou plus autres noeuds ancêtres, jusqu'à ce que ledit nombre suffisant de noeuds destinés à remplir la fenêtre d'affichage (132A, 132B) a été déterminé.

3. Le support selon la Revendication 1, où ladite liste ordonnée de noeuds comprend un ou plusieurs noeuds ancêtres du noeud cible, et où lesdits un ou plusieurs noeuds ancêtres du noeud cible sont déterminés par un parcours vers le haut à partir du noeud cible dans l'arbre hiérarchique de noeuds placé dans une mémoire tampon locale (122).

4. Le support selon l'une quelconque des Revendications 1 à 3, où ledit nombre suffisant de noeuds destinés à remplir la fenêtre d'affichage (132A, 132B) est déterminé en fonction de la configuration correspondant à la fenêtre d'affichage (132A, 132B), et où la configuration correspondant à la fenêtre d'affichage (132A, 132B) comprend un ou plusieurs éléments parmi : une taille de la fenêtre d'affichage, une résolution de la fenêtre d'affichage (132A, 132B), une taille d'illustrations de noeud et une quantité d'espa-

ce entre des illustrations de noeud.

5. Le support selon l'une quelconque des Revendications 1 à 4, où la mise en mémoire tampon d'un noeud particulier des un ou plusieurs noeuds extraits de l'ordinateur distant (112) comprend en outre :
le réglage de la valeur d'un indicateur d'état d'expansion pour le noeud particulier sur une valeur par défaut, où :

une valeur qui indique que l'état du noeud particulier est étendu amène des noeuds enfants du noeud particulier à être affichés de manière contiguë au noeud particulier, et
une valeur qui indique que l'état du noeud particulier est replié amène le noeud particulier à être affiché sans l'affichage de noeuds enfants du noeud particulier.

6. Le support selon la Revendication 5, où les instructions amènent en outre :

la réception d'une demande d'expansion de la totalité des noeuds dans l'arbre hiérarchique de noeuds (122),
en réponse à la réception de la demande d'expansion de la totalité des noeuds dans l'arbre hiérarchique de noeuds (122), le réglage d'une valeur d'un indicateur d'état d'expansion globale destinée à indiquer que la totalité des noeuds représentant des noeuds dans l'arbre hiérarchique de noeuds (122) sont à afficher sous la forme étendue,
la détermination, lorsque la demande est reçue, qu'un noeud affiché est affiché et que des noeuds enfants du noeud affiché ne sont pas affichés,
l'extraction d'un ou de plusieurs noeuds enfants du noeud affiché qui sont absents de la mémoire tampon sans l'extraction de noeuds enfants d'autres noeuds qui ne sont pas affichés,
la modification de la liste de noeuds affichée de façon à inclure une partie des noeuds enfants du noeud particulier, et
l'ajout des un ou plusieurs noeuds enfants extraits à la hiérarchie de noeuds placée en mémoire tampon (144).

7. Le support selon la Revendication 5, où l'ordinateur distant (112) répond à des demandes relatives à des noeuds dans l'arbre hiérarchique de noeuds (122) sans effectuer un suivi :

(a) des noeuds qui ont été envoyés au demandeur ou
(b) de l'état de la hiérarchie de noeuds placée en mémoire tampon (144) entretenue par le demandeur.

8. Le support selon l'une quelconque des Revendications 1 à 7, où :

chaque noeud dans l'arbre hiérarchique de noeuds (122) possède une séquence ordonnée de zéro ou plus noeuds enfants, où un ou plusieurs noeuds enfants de la séquence ordonnée de noeuds enfants sont affichés dans l'ordre de leur séquence,
des noeuds enfants de noeuds non étendus sont omis de la liste ordonnée de noeuds, et
la détermination de la liste ordonnée de noeuds comprend en outre la sélection de noeuds contigus dans la liste ordonnée de noeuds, où :

les noeuds contigus sélectionnés comprennent le noeud cible, et
le nombre de noeuds contigus est basé sur le nombre de noeuds qui peuvent être affichés conjointement.

9. Le support selon l'une quelconque des Revendications 1 à 8, où l'extraction comprend en outre l'extraction d'un autre sous-ensemble de noeuds qui sont adjacents au deuxième sous-ensemble de noeuds dans l'arbre hiérarchique de noeuds (122), où l'autre sous-ensemble de noeuds n'est pas affiché avec le noeud cible.

10. Un procédé comprenant les opérations décrites dans l'une quelconque des Revendications 1 à 9.

11. Un système comprenant :

au moins un dispositif matériel comprenant un processeur (304), et
le système étant configuré de façon à exécuter les opérations décrites dans l'une quelconque des Revendications 1 à 9.

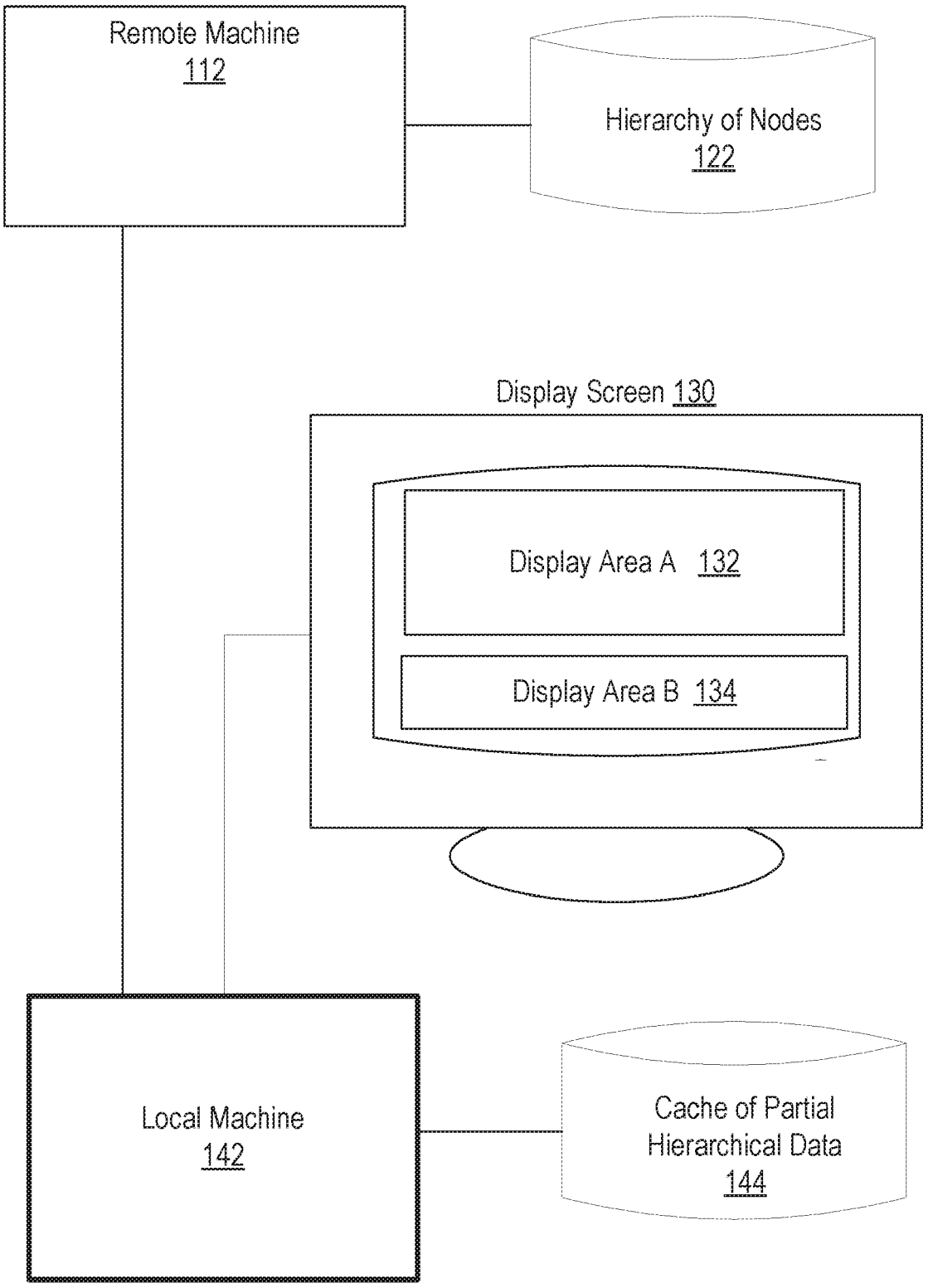

Remote Machine
112
Hierarchy of Nodes
122
Display Screen 130
Display Area A 132
Display Area B 134
Local Machine
142
Cache of Partial
Hierarchical Data
144

FIG. 1A

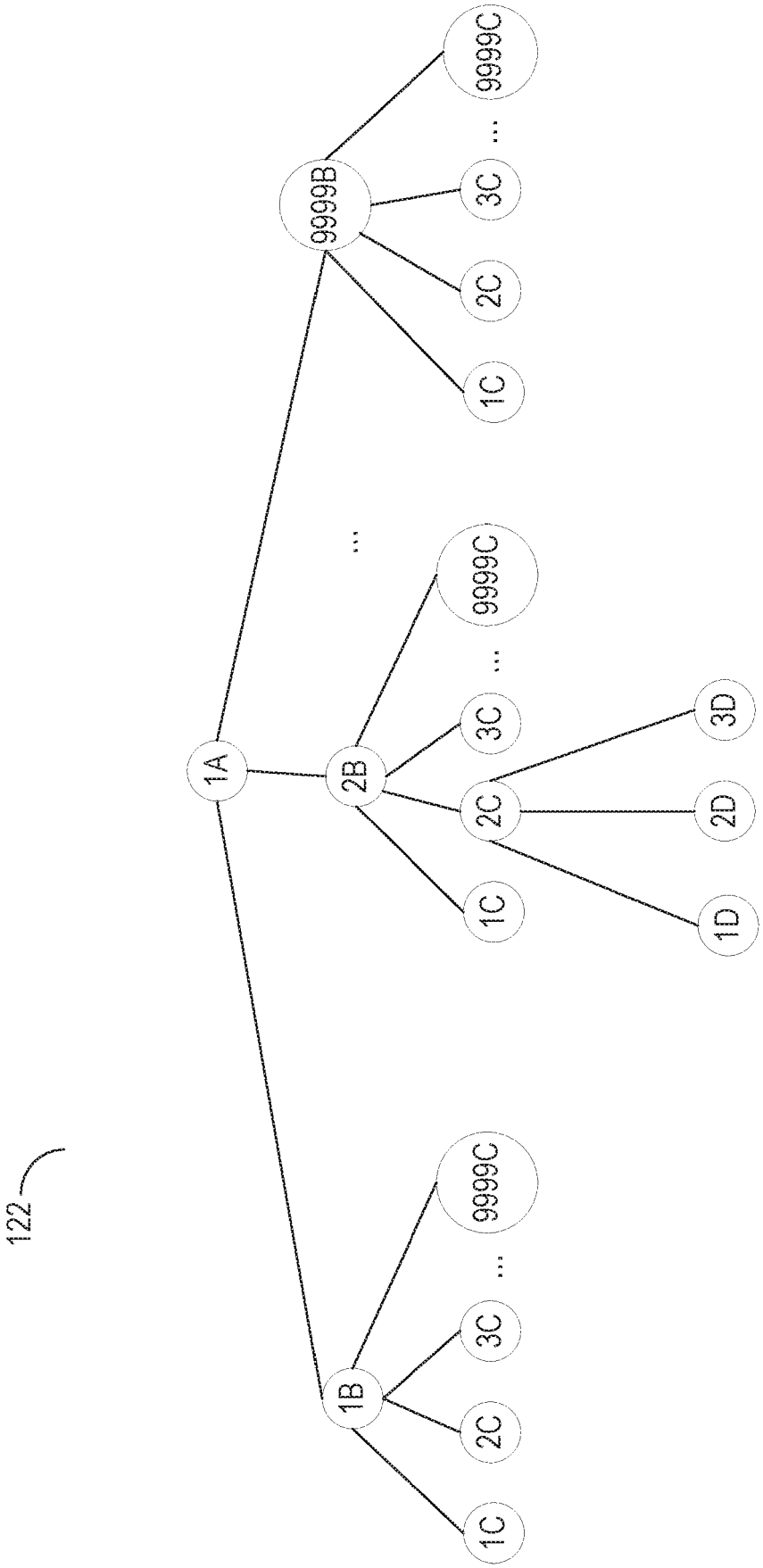

122
1A
1B
1C
2C
3C
9999C
2B
1C
2C
3C
9999C
3D
2D
1D
9999B
1C
2C
3C
9999C

FIG. 1B

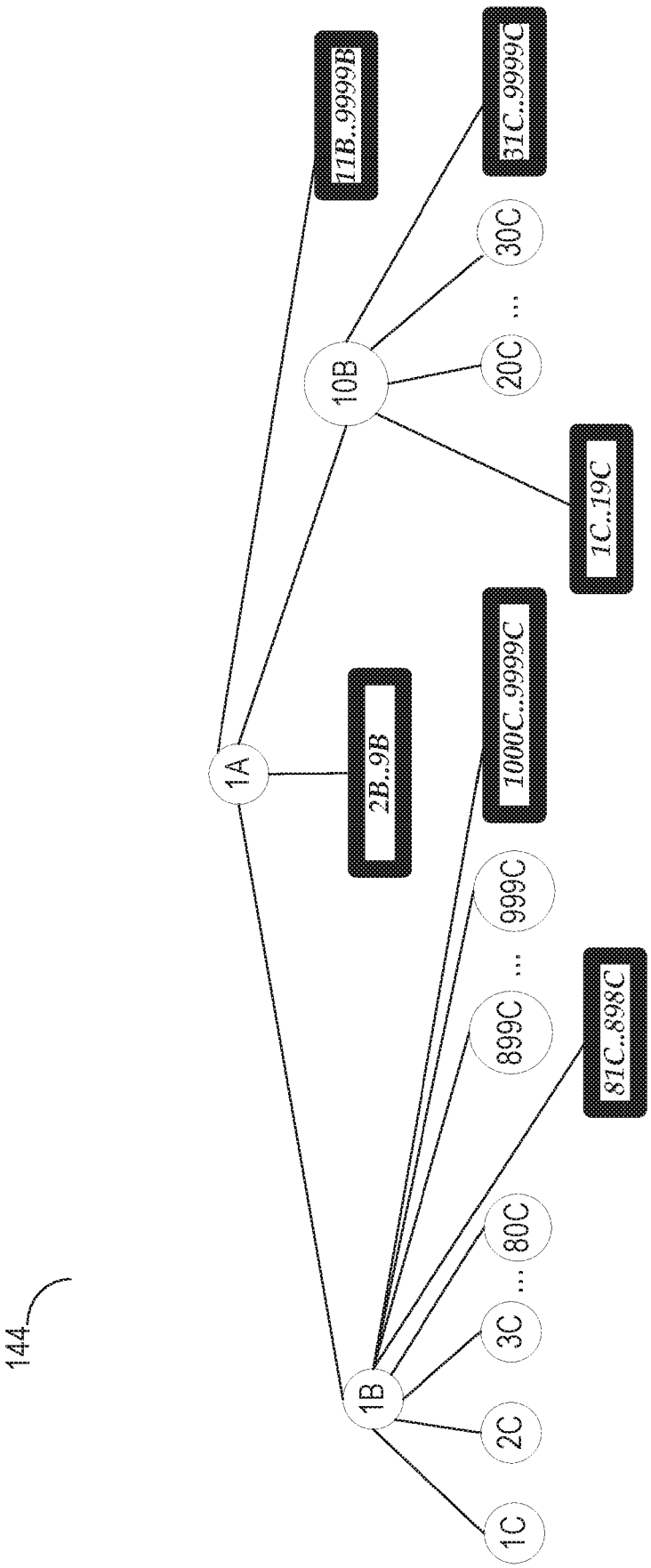

144
1A
2B..9B
11B..9999B
10B
1C..19C
20C ...
30C
31C..9999C
1B
1C
2C
3C ...
80C
81C..898C
899C ...
999C
1000C..9999C

FIG. 1C

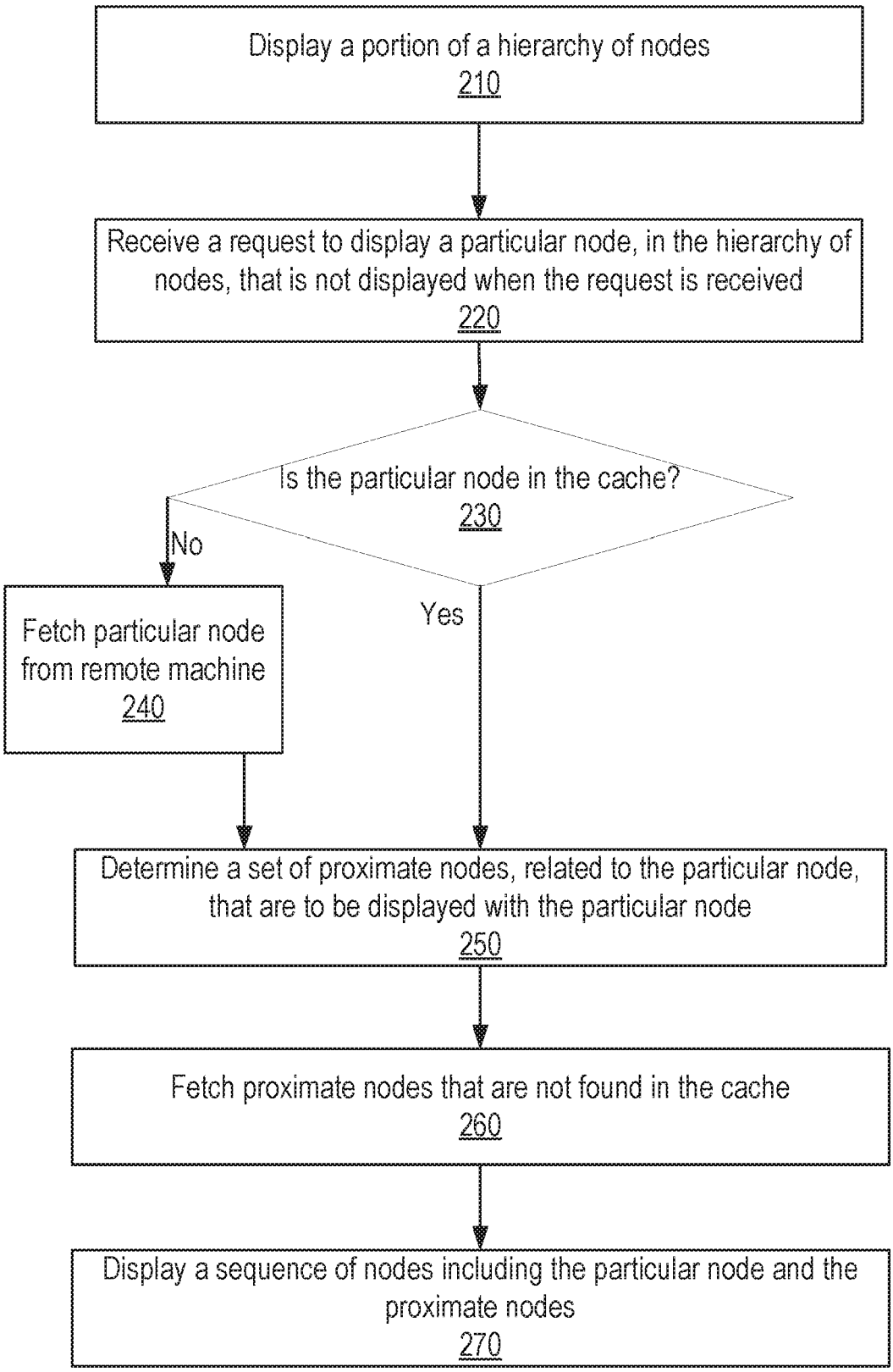
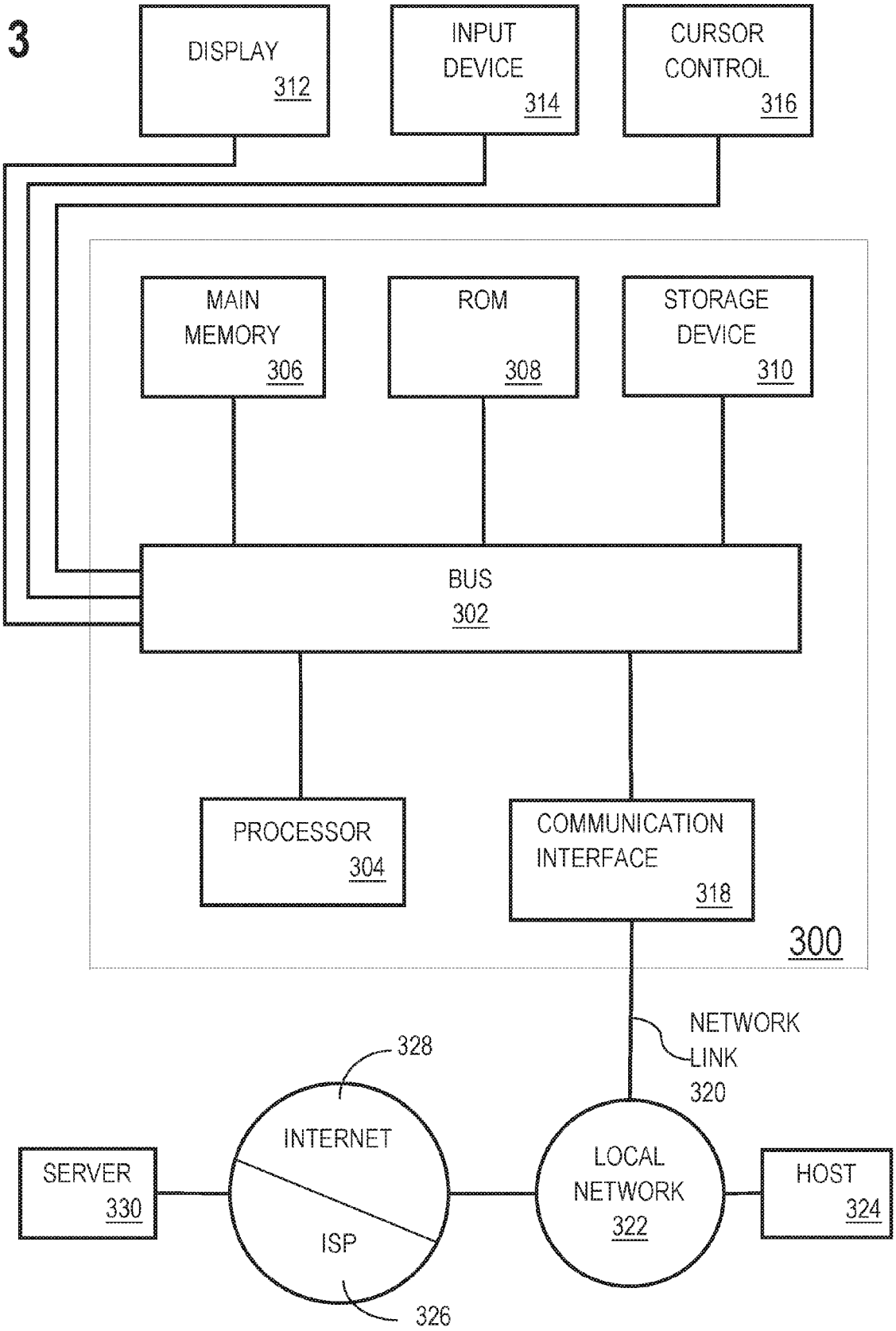

Display a portion of a hierarchy of nodes
210
Receive a request to display a particular node, in the hierarchy of nodes, that is not displayed when the request is received
220
Is the particular node in the cache?
230
No
Yes
Fetch particular node from remote machine
240
Determine a set of proximate nodes, related to the particular node, that are to be displayed with the particular node
250
Fetch proximate nodes that are not found in the cache
260
Display a sequence of nodes including the particular node and the proximate nodes
270

FIG. 2

FIG. 3


DISPLAY
312
INPUT
DEVICE
314
CURSOR
CONTROL
316
MAIN
MEMORY
306
ROM
308
STORAGE
DEVICE
310
BUS
302
PROCESSOR
304
COMMUNICATION
INTERFACE
318
300
NETWORK
LINK
320
328
INTERNET
ISP
326
SERVER
330
LOCAL
NETWORK
322
HOST
324

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

- US 20160321287 A1 **[0005]**