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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
9 March 2017 (09.03.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/039703 A1

(51) International Patent Classification:
G06F 17/30 (2006.01) *G06F 17/00* (2006.01)

(21) International Application Number:
PCT/US2015/048728

(22) International Filing Date:
4 September 2015 (04.09.2015)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP** [US/US]; 11445 Compaq Center Drive West, Houston, Texas 77070 (US).

(72) Inventors: **MARWAH, Manish**; 1501 Page Mill Rd., Palo Alto, California 84304 (US). **KIM, Mijung**; 1501 Page Mill Rd., Palo Alto, California 94304 (US).

(74) Agents: **HARTMANN, Kenneth R.** et al.; Hewlett Packard Enterprise, 3404 E. Harmony Road, Mail Stop 79, Fort Collins, Colorado 80528 (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, BN, 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, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, 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, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, 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, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to the identity of the inventor (Rule 4.17(i))*

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

Published:

— *with international search report (Art. 21(3))*

(54) Title: HYBRID GRAPH PROCESSING

(57) Abstract: Examples herein involve identifying characteristics associated with nodes of a graph, selecting a set of nodes from the nodes to be processed based on hybrid graph processing settings and the characteristics, and executing, via a processor, the selected set of nodes in parallel to process the graph.

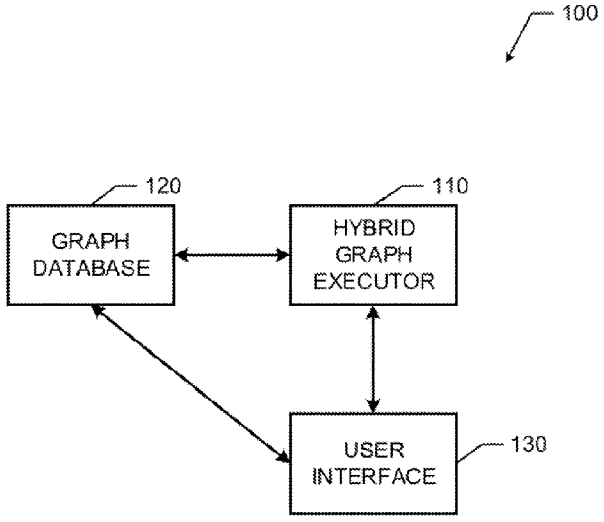

FIG. 1

WO 2017/039703 A1

HYBRID GRAPH PROCESSING

BACKGROUND

[0001] A graph is a representation of a set of data (e.g., Big Data). An example graph may include a plurality of nodes (e.g., executables) and edges connecting the plurality of edges. The graph may be processed by executing nodes in accordance with characteristics of edges linked to the nodes and related nodes linked by the edges.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] FIG. 1 is a block diagram of an example graph processing system including a hybrid graph executor that may be implemented in accordance with an aspect of this disclosure.

[0003] FIG. 2 is a block diagram of an example hybrid graph executor that may be used to implement the hybrid graph executor of FIG. 1.

[0004] FIG. 3 is a flowchart representative of example machine readable instructions that may be executed to implement the hybrid graph executor of FIG. 2.

[0005] FIG. 4 is a flowchart representative of an example portion of the example machine readable instructions of FIG. 3 to implement a node selector of the hybrid graph executor of FIG. 2.

[0006] FIG. 5 is another flowchart representative of an example portion of the example machine readable instructions of FIG. 3 to implement a node selector of the hybrid graph executor of FIG. 2.

[0007] FIG. 6 is yet another flowchart representative of an example portion of the example machine readable instructions of FIG. 3 to implement a node selector of the hybrid graph executor of FIG. 2.

[0008] FIG. 7 is a block diagram of an example processor platform capable of executing the instructions of FIGS. 3, 4, 5, and/or 6 to implement the hybrid graph executor of FIG. 2.

[0009] Wherever possible, the same reference numbers will be used throughout the drawing(s) and accompanying written description to refer to the same or like parts.

DETAILED DESCRIPTION

[0010] Examples disclosed herein involve parallel graph processing that combines a synchronous execution model and an asynchronous execution model to process graphs. Example graphs herein are data representations of large data sets (which may be referred to as "Big Data"). In examples herein, adjustments may be made to a hybrid graph executor to control a degree of synchronous execution versus asynchronous execution when processing graphs. Accordingly, examples herein provide a hybrid synchronous/asynchronous approach to processing graphs.

[0011] In synchronous graph processing, a plurality of nodes may be processed in parallel in a plurality of iterations. Synchronous graph processing may be relatively simple to implement and is advantageous in its ability to process multiple nodes at the same time in a single iteration. However, some issues may arise as processed nodes may not be properly updated for each iteration as parallelization does not necessarily account for execution order of the nodes. Accordingly, convergence of processing the graph may be slowed due to processing outdated nodes, outdated messages of edges, or other potential outdated characteristics. In asynchronous graph processing, there are no iterations and nodes are processed based on priority. In such examples, the nodes with highest priority may be the most up to date as they were the most recently processed. However, locking mechanisms may need to be implemented in asynchronous implementations as contention arises between nodes sending and receiving messages to the order of execution of the nodes. Examples herein, provide for parallel processing of sets of nodes selected

based on characteristics of the nodes of the graph. Accordingly, examples herein provide a hybrid synchronous/asynchronous graph processing technique that may be adjustable based on user input, graph type, processing type, etc. that enables parallel processing of sets of nodes selected in each iteration based on priorities (or updated priorities) of the nodes of the graph.

[0012] Examples herein take advantage of the benefits of both synchronous execution and asynchronous execution by combining the models into a hybrid graph process execution. Further, the disadvantages corresponding to synchronous execution and asynchronous may be mitigated by adjusting settings to control the degree of synchronous execution versus asynchronous execution in accordance with the teachings of this disclosure.

[0013] An example method includes identifying characteristics associated with nodes of a graph; selecting a set of nodes from the nodes to be processed based on hybrid graph processing settings and the characteristics; and executing, via a processor, the selected set of nodes in parallel to process the graph.

[0014] As used herein, graph processing (or processing a graph) involves processing nodes of a graph until all nodes (or a threshold percentage of the nodes) have been processed, which is known as convergence of the graph processing. Convergence occurs when all nodes (or a threshold percentage of the nodes) have been processed such that a particular characteristic of the nodes has reached a fixed point (or value), and, thus, does not change with subsequent iterations. In examples herein, parallel graph processing or processing graphs in parallel may involve several iterations of processing a set of nodes in parallel (at a same time or within a same time period). For example, in a first iteration, a first set of nodes may be processed and in a second iteration a second set of nodes may be processed. In examples herein, the second set of nodes may or may not include nodes related or linked to the first set of nodes.

[0015] As used herein, priority or priority of (or associated with) a node refers to an importance that the node be processed sooner rather than later. For example, it is desirable to have that a node having a higher priority is

processed before a node having a lower priority. Priority may be ranked or rated on any suitable scale, (e.g., from 1 to 10, 10 being highest priority, 1 being lowest priority (or vice versa).

[0016] FIG. 1 is a block diagram of an example graph processing system 100 including a hybrid graph executor 110 constructed in accordance with an aspect of this disclosure. The example graph processing system 100 includes the hybrid graph executor 110, a graph database 120, and a user interface 130. In the example of FIG. 1, the hybrid graph executor 110 processes graphs received from the graph database 120 and provides results of the processed graph to the user interface 130.

[0017] The example graph database 120 of FIG. 1 stores graphs and may provide graphs to the hybrid graph executor 110 for processing. In some examples, the graph database 120 of FIG. 1 may include a controller of the database that provides the graphs to the hybrid graph executor 110. For example, the graph database 120 may be a local storage device, a network storage device (e.g., a server), etc. that stores large data sets (e.g., which may be referred to as "Big Data") represented by graphs and/or the graphs themselves. The example graph database 120 may include a storage device (e.g., a non-volatile storage, a flash memory) and/or a memory device (e.g., a dynamic random access memory (DRAM), a non-volatile random access memory (NVRAM)). Additionally, or alternatively, the hybrid graph executor 110 may retrieve graphs from the graph database 120.

[0018] The example user interface 130 facilitates user interaction with the hybrid graph executor 110 and/or the graph database 120. For example, the user interface may include user input(s) (e.g., a keyboard, a mouse, a trackball, a touchscreen, etc.) and/or user output(s) (e.g., a display, a touchscreen, speakers, indicators, etc.). In examples herein, the user input 130 enables a user to access and control settings of the hybrid graph executor 110. For example, the user interface 130 may request processing of a particular graph to be provided by the graph database 120. Furthermore, the user interface 130 enables a user to indicate or set hybrid graph processing settings to process the selected graph in accordance with examples herein.

[0019] The example hybrid graph executor 110 processes graphs in accordance with aspects of this disclosure. The hybrid graph executor 110 processes a number of nodes of a graph (e.g., in parallel) in accordance with aspects of this disclosure. For example, the hybrid graph executor 110 may determine a number of the nodes of the graph to be processed based on user settings and/or characteristics of the graph data (or the graph processing system 100 of FIG. 1).

[0020] FIG. 2 is a block diagram of an example hybrid graph executor 110 that may be used to implement the hybrid graph executor 110 of FIG. 1. The example hybrid graph executor 110 of FIG. 2 includes a graph receiver 210, a settings manager 220, a node selector 230, and a hybrid execution engine 240. In examples herein, the graph receiver 210 receives a graph (or a plurality of graphs) and the node selector 230 selects nodes of the graph for execution by the hybrid execution engine 240 based on settings of the settings manager 220 and characteristics of the nodes. The example hybrid execution engine 240 iteratively processes sets of nodes of the graphs in parallel and may adjust characteristics of the nodes (e.g., priorities) based on the execution. Accordingly, after each parallel processing of a set of nodes (which may be referred to herein as an iteration), the hybrid execution engine 240 may provide feedback to the node selector 230 by adjusting characteristics of the nodes for consideration by node selector 230 when selecting nodes for a subsequent iteration.

[0021] The example graph receiver 210 receives (or retrieves) graph data of a graph from the graph database 120 of FIG. 1. The example graph data includes node data corresponding to nodes of the graph and edge data corresponding to edges of the graph. In some examples, the graph receiver 210 may include or serve as a buffer (or other short term memory device/data structure) and the hybrid graph executor 110 may load the graph data into the graph receiver 210 for processing. Accordingly, the graph receiver 210 may store the graph data for a graph while the node selector 230 and/or the hybrid execution engine process the graph data. In some examples, the graph data may be separated by the graph receiver into node data and edge data.

Accordingly, the graph receiver 210 may include a data structure that allows the node selector 230 to identify or access the node data for nodes of the graph in accordance with examples herein.

[0022] The example settings manager 220 manages hybrid graph processing settings for the hybrid graph executor 110 to determine how a particular graph is to be processed. In examples herein, the hybrid graph processing settings may be adjusted or tuned to adjust the degree of synchronous processing versus asynchronous processing when performing a hybrid graph processing in accordance with examples herein. The settings manager 220 may include default settings or settings established in response to user input (e.g., received via the user interface 130). The hybrid graph processing settings may include a selection type and sub-settings for that particular selection type. For example, selection types may include a selection of nodes having a minimum priority (or certain set of characteristics), selection of a threshold number of nodes for processing, or selection of bins of nodes based on the priority of the nodes. Based on the settings of the graph executor 110, the settings manager 220 may provide instructions to the node selector 230 for selecting nodes. In some examples, based on a selection type, the settings manager 220 may instruct the node selector 230 to use a particular data structure (e.g., a priority queue, bins, randomizer, etc.) to select nodes. Furthermore, the sub-settings may include settings specific to the selection type, such as a setting for selecting (or establishing) a threshold minimum priority, a setting for selecting (or establishing) a minimum number of nodes to select (e.g., based on priority), or settings for numbers of bins, a minimum priority of nodes in the bins to be selected, and/or a range of priorities for each bin. Accordingly, the settings manager 220 may provide settings or instruct the node selector 230 to select nodes from the graph data based on the settings (e.g., default settings, settings received via user input, etc.). In some examples, the settings manager 220 may include or may communicate with a graphical user interface to facilitate interaction with a user via the user interface 130.

[0023] The example node selector 230 analyzes node data and selects nodes from the node data for processing/execution by the hybrid execution

engine 240 based on the settings of the settings manager 220. The example node selector 230 may include any suitable data structure for selecting nodes to be processed for each iteration of processing a graph. For example, the node selector 230 may include or implement a priority queue that sorts retrieved/received nodes from the graph receiver 210 and/or from the hybrid execution engine 240. The node selector 230 may then select a number of nodes from the priority queue based on the graph processing settings. In some examples, the node selector 130 may select all nodes from the priority queue with a priority that satisfies a threshold priority. For example, the graph processing settings from the settings manager 220 may indicate that all nodes having a priority of '5' or higher be selected or that all nodes having a priority of 3 or lower be selected, etc. In such examples, the node selector 230 may easily identify the threshold priority from the priority queue and select the nodes from either side (e.g., a side having higher priority nodes and a side having lower priority nodes) of the priority queue. In some examples, for an initial iteration of processing a graph, the node selector 230 may randomly select nodes or randomly assign a priority (or a same priority) to the nodes. The example random priority may be adjusted during processing by the hybrid execution engine 240 as discussed below.

[0024] In some examples, the node selector 230 may select a threshold number of nodes from an end of the priority queue based on the graph processing settings. For example, the graph processing settings may indicate that the "top 10" nodes are to be selected for each iteration of processing or that the "bottom 5" nodes are to be selected for each iteration of processing. Accordingly, for a selected set of nodes, the threshold number may correspond to a magnitude of the selected set of nodes (i.e., the amount or number of nodes in the selected set). Using this selection type, the settings manager 220 and/or node selector 230 may adjust a degree of synchronous versus asynchronous processing. For example, for a graph having X nodes, and N being equal to a threshold number nodes to be executed in an iteration of the processing, then when N = X, the hybrid graph executor 110 may synchronously process the graph and when N = 1 the hybrid graph executor 110 may

asynchronously process the graph. Accordingly, the hybrid graph processing settings may allow for adjusting a degree of synchronous versus asynchronous processing of the graph from fully synchronous to fully asynchronous.

[0025] In some examples, the node selector 230 may utilize a data structure that includes a plurality of bins. The bins may store a predetermined or unlimited number of nodes to be selected for processing. The node selector 230 may select nodes to be processed based on bin assignments. In examples herein, the nodes may be assigned to the bins based on the characteristics of the nodes (e.g., node type, execution type, priority, etc.). In some examples, the node selector 230 may assign the bins using hashing, such that the characteristic(s) of the nodes may be used as a hash key. For example, the bins may be arranged to temporarily store the nodes based on priority, such that the bins temporarily store the nodes from a lowest priority to a highest priority (or vice versa). In such examples, the node selector 230 may select nodes by selecting the nodes from a number of the bins based on the graph processing settings. For example, the graph processing settings may indicate that a threshold number of bins (e.g., 2 bins, 4 bins, etc.) including nodes with a higher priority (or nodes with a lower priority) are to be selected. In some examples, the graph processing settings may indicate that nodes from bins including a node that satisfies a threshold priority (e.g., “5 or higher”, “3 or lower”, etc.) are to be selected. Accordingly, in such an example, all nodes from the bin may be selected regardless of whether that particular node has a priority that satisfies the bin threshold priority.

[0026] Accordingly, in examples herein, for each iteration of processing the graph, the node selector 230 selects nodes for execution by the hybrid execution engine 240 based on characteristics of the nodes (e.g., node type, priority, execution time, etc.). The example hybrid execution engine 240 processes/executes the selected nodes in parallel with one another (e.g., similar to a synchronous processing model). For example, for each iteration the hybrid execution engine 240 may process the nodes in a series of corresponding threads by processing data/messages of edges of the respective nodes of that iteration. After/while processing the nodes, the hybrid execution engine 240

may assign priorities to (or update priorities of) the selected nodes after they are processed. In some examples, the hybrid execution engine 240 may assign priorities to (or update priorities of) nodes linked or connected to the selected nodes by edges of the graph based on execution status that indicates execution time (length of execution or remaining execution time until convergence), node type, distance from convergence of node, etc. For example, the hybrid execution engine 240 may lower a priority of a node that has been processed, has a lower processing demand (or execution time), is not to be processed again, and the hybrid execution engine 240 may increase a priority of a node if the node is to be processed next (e.g., has a greater execution urgency, such as a node that is linked to a selected node), a node that requires a relatively long execution time (e.g., a relatively large change has occurred in a characteristic of the node during an iteration, a node linked to a relatively large number of other nodes), etc. The hybrid execution engine 240 may then feed the processed nodes with updated priorities and priority data corresponding to linked nodes back to the node selector 230. The example node selector 230 may then update a data structure (e.g., the priority queue, bins, etc.) by rearranging, reorganizing, or re-sorting the nodes for a subsequent selection/iteration to process the graph.

[0027] The example hybrid execution engine 240 may check for convergence of the processed graph by tracking that all nodes of the graph (or a threshold percentage of the nodes) have been processed. Accordingly, once all iterations are complete, the hybrid execution engine 240 may output the results of the graph processing (e.g., to the user interface 130 or back to the graph database 120).

[0028] Accordingly, the example hybrid graph executor 110 of FIG. 2 may process graphs using both synchronous (using the hybrid graph execution engine to process selected nodes in parallel) and asynchronous models (using the settings manager 220 and node selector 230 to select sets of nodes for iterations based on characteristics of the nodes).

[0029] While an example manner of implementing the hybrid graph executor 110 of FIG. 1 is illustrated in FIG. 2, at least one of the elements,

processes and/or devices illustrated in FIG. 2 may be combined, divided, re-arranged, omitted, eliminated and/or implemented in any other way. Further, the graph receiver 210, the settings manager 220, the node selector 230, the hybrid graph execution engine 240, and/or, more generally, the hybrid graph executor 110 of FIG. 2 may be implemented by hardware and/or any combination of hardware and executable instructions (e.g., software and/or firmware). Thus, for example, any of the graph receiver 210, the settings manager 220, the node selector 230, the hybrid graph execution engine 240, and/or, more generally, the hybrid graph executor 110 could be implemented by at least one of an analog or digital circuit, a logic circuit, a programmable processor, an application specific integrated circuit (ASIC), a programmable logic device (PLD) and/or a field programmable logic device (FPLD). When reading any of the apparatus or system claims of this patent to cover a purely software and/or firmware implementation, at least one of the graph receiver 210, the settings manager 220, the node selector 230, and/or the hybrid graph execution engine 240 is/are hereby expressly defined to include a tangible machine readable storage device or storage disk such as a memory, a digital versatile disk (DVD), a compact disk (CD), a Blu-ray disk, etc. storing the executable instructions. Further still, the example hybrid graph executor 110 of FIG. 2 may include at least one element, process, and/or device in addition to, or instead of, those illustrated in FIG. 2, and/or may include more than one of any or all of the illustrated elements, processes and devices.

[0030] Flowcharts representative of example machine readable instructions for implementing the hybrid graph executor 110 of FIG. 2 are shown in FIGS. 3, 4, 5, 6 and 7. In this example, the machine readable instructions comprise program(s)/process(es) for execution by a processor such as the processor 812 shown in the example processor platform 800 discussed below in connection with FIG. 8. The program(s)/process(es) may be embodied in executable instructions (e.g., software) stored on a tangible machine readable storage medium such as a CD-ROM, a floppy disk, a hard drive, a digital versatile disk (DVD), a Blu-ray disk, or a memory associated with the processor 812, but the entire program/process and/or parts thereof could alternatively be

executed by a device other than the processor 812 and/or embodied in firmware or dedicated hardware. Further, although the example program(s) is/are described with reference to the flowchart illustrated in FIGS. 3, 4, 5, 6, and/or 7, many other methods of implementing the example hybrid graph executor 110 may alternatively be used. For example, the order of execution of the blocks may be changed, and/or some of the blocks described may be changed, eliminated, or combined.

[0031] The example process 300 of FIG. 3 begins with an initiation of the hybrid graph executor 110 (e.g., upon startup, upon instructions from a user, upon startup of a device implementing the hybrid graph executor 110 (e.g., the graph processing system 100), etc.). The example process 300 may be executed to process a graph in a hybrid synchronous/asynchronous manner. Furthermore, the example process 300 may be iteratively executed to process the graph. At block 310, the node selector 230 identifies characteristics associated with nodes of a graph. For example, at block 310, the node selector 230 may identify priorities, node types, etc. of the nodes of the graph.

[0032] At block 320, the node selector 230 selects a set of nodes to be processed based on graph processing settings and the characteristics of the nodes. For example, the node selector 230 may use a selection type and sub-settings from a settings manager to select nodes for execution (in the iteration of FIG. 3). Examples processes to select the nodes are further discussed below in connection with FIGS. 4, 5, and 6. At block 330, the hybrid execution engine executes the set of nodes in parallel to process the graph. After block 330, the example process 300 ends. In some examples, after block 330, the hybrid execution engine 240 may determine whether graph processing is complete. If graph processing isn't complete (e.g., all nodes, or a threshold percentage of the nodes, have not converged), then control may return to block 310. If the graph processing has completed (the processed nodes have converged), then the example process 300 ends.

[0033] The example process 400 of FIG. 4 begins with an initiation of the node selector 230 (e.g., in response to receiving a graph data, in response to receiving updated node data from the hybrid execution engine, in response to

instructions from the settings manager and/or user interface 130, etc.). The example process 400 of FIG. 4 may be executed to implement the example block 320 of FIG. 3 to select nodes that satisfy a threshold priority based on the hybrid graph settings of the settings manager 220.

[0034] At block 410 of FIG. 4, the node selector 230 sorts nodes of the graph by priority. For example, the node selector 230 may use a priority queue to sort the nodes from lowest priority to highest priority (or vice versa). At block 420, the node selector 230 identifies a threshold priority indicated or maintained by the settings manager 220 (e.g., in response to a user input). At block 430, the node selector 230 selects nodes from a priority queue of the node selector 230 based on the threshold priority. The example priority queue of the node selector 230 may sort the nodes based on priority level. For example, at block 430, the node selector 230 may scan the priority queue for a node that has the identified priority, select that node and any other nodes on either side of the priority queue (based on whether the threshold is a greater than or less than threshold). At block 440, the node selector 230 provides the selected nodes to the hybrid execution engine 240 to be processed. After block 440, the example process 400 ends. In some examples, after the example process 400, control may advance to block 330 of FIG. 3.

[0035] The example process 500 of FIG. 5 begins with an initiation of the node selector 230. The example process 500 of FIG. 5 may be executed to implement the example block 320 of FIG. 3 to select a threshold number of nodes based on the hybrid graph settings of the settings manager 220.

[0036] At block 510 of FIG. 6, the node selector 230 sorts nodes of the graph by priority. For example, the node selector 230 may use a priority queue to sort the nodes from lowest priority to highest priority (or vice versa). At block 520, the node selector 230 selects a threshold number (e.g., a magnitude of a selected set of nodes) of nodes from the sorted nodes (e.g., from a higher priority end of the priority queue, or from a lower priority end of the priority queue). For example, the node selector 230, at block 520, may retrieve or receive hybrid graph processing settings indicating that the threshold number (e.g., 10, 15, etc.) of nodes are to be processed for each iteration. At block 530,

the node selector 230 provides the selected nodes to the hybrid execution engine 240 to be processed. After block 530, the example process 400 ends. In some examples, after the example process 400, control may advance to block 330 of FIG. 3.

[0037] The example process 600 of FIG. 6 begins with an initiation of the node selector 230. The example process 600 of FIG. 6 may be executed to implement the example block 320 of FIG. 3 to select a threshold number of nodes based on the hybrid graph settings of the settings manager 220.

[0038] At block 610 of FIG. 6, the node selector 230 designates a plurality of bins to be used in a data structure of the node selector 230 based on the hybrid graph settings. At block 620, the node selector 230 assigns the nodes to the bins based on characteristics of the nodes (e.g., based on priorities of the nodes, using a hashing technique, etc.). In some examples, the node selector 230 may evenly divide the nodes among the plurality of bins or assign threshold priorities to the plurality of bins. At block 630, the node selector 230 selects the nodes from a threshold number (e.g., 3, 5, etc.) of bins based on the hybrid graph processing settings. For example, the hybrid graph processing settings may indicate that nodes from the top 3 (or bottom 3) bins are to be selected. At block 640, the node selector 230 provides the selected nodes to the hybrid execution engine 240 to be processed. After block 640, the example process 400 ends. In some examples, after the example process 400, control may advance to block 330 of FIG. 3.

[0039] The example process 700 of FIG. 7 begins with an initiation of the hybrid execution engine 240 (e.g., in response to receiving nodes from the node selector 230). The example process 700 of FIG. 7 may be executed to implement the example block 330 of FIG. 3 to execute selected nodes in parallel and provide updated priorities to the node selector 230 based on the execution of the nodes.

[0040] At block 710 of FIG. 7, the hybrid graph execution engine 240 executes the selected nodes in parallel (e.g., in a synchronous manner). At block 720, the hybrid execution engine 240 determines/calculates updated priorities of the selected nodes and/or nodes linked to the selected nodes (e.g.,

which may be identified in processed edges of the selected nodes). The example hybrid execution engine 240 provides the updated priorities to the node selector 230 to be used for subsequent iterations of the graph processing. Accordingly, after block 730, the node selector 230 may reorganize or re-sort the nodes (e.g., in a priority queue, in bins, etc.) for a subsequent iteration. After block 730, the example process 700 ends.

[0041] As mentioned above, the example processes of FIGS. 3, 4, 5, 6, and/or 7 may be implemented using coded instructions (e.g., computer and/or machine readable instructions) stored on a tangible machine readable storage medium such as a hard disk drive, a flash memory, a read-only memory (ROM), a compact disk (CD), a digital versatile disk (DVD), a cache, a random-access memory (RAM) and/or any other storage device or storage disk in which information is stored for any duration (e.g., for extended time periods, permanently, for brief instances, for temporarily buffering, and/or for caching of the information). As used herein, the term tangible machine readable storage medium is expressly defined to include any type of machine readable storage device and/or storage disk and to exclude propagating signals and to exclude transmission media. As used herein, "computer readable storage medium" and "machine readable storage medium" are used interchangeably. Additionally or alternatively, the example processes of FIGS. 3, 4, 5, 6, and/or 7 may be implemented using coded instructions (e.g., computer and/or machine readable instructions) stored on a non-transitory computer and/or machine readable medium such as a hard disk drive, a flash memory, a read-only memory, a compact disk, a digital versatile disk, a cache, a random-access memory and/or any other storage device or storage disk in which information is stored for any duration (e.g., for extended time periods, permanently, for brief instances, for temporarily buffering, and/or for caching of the information).

[0042] As used herein, the term non-transitory machine readable medium is expressly defined to include any type of machine readable storage device and/or storage disk and to exclude propagating signals and to exclude transmission media. As used herein, when the phrase "at least" is used as the transition term in a preamble of a claim, it is open-ended in the same manner as

the term "comprising" is open ended. As used herein the term “a” or “an” may mean “at least one,” and therefore, “a” or “an” do not necessarily limit a particular element to a single element when used to describe the element. As used herein, when the term “or” is used in a series, it is not, unless otherwise indicated, considered an “exclusive or.”

[0043] FIG. 8 is a block diagram of an example processor platform 800 capable of executing the instructions of FIGS. 3, 4, 5, 6, and/or 7 to implement the hybrid graph executor of FIG. 2. The example processor platform 800 may be or may be included in any type of apparatus, such as a server, a personal computer, a mobile device (e.g., a cell phone, a smart phone, a tablet, etc.), a personal digital assistant (PDA), an Internet appliance, any other type of computing device.

[0044] The processor platform 800 of the illustrated example of FIG. 8 includes a processor 812. The processor 812 of the illustrated example is hardware. For example, the processor 812 can be implemented by at least one integrated circuit, logic circuit, microprocessor or controller from any desired family or manufacturer.

[0045] The processor 812 of the illustrated example includes a local memory 813 (e.g., a cache). The processor 812 of the illustrated example is in communication with a main memory including a volatile memory 814 and a non-volatile memory 816 via a bus 818. The volatile memory 814 may be implemented by Synchronous Dynamic Random Access Memory (SDRAM), Dynamic Random Access Memory (DRAM), RAMBUS Dynamic Random Access Memory (RDRAM) and/or any other type of random access memory device. The non-volatile memory 816 may be implemented by flash memory and/or any other desired type of memory device. Access to the main memory 814, 816 is controlled by a memory controller.

[0046] The processor platform 800 of the illustrated example also includes an interface circuit 820. The interface circuit 820 may be implemented by any type of interface standard, such as an Ethernet interface, a universal serial bus (USB), and/or a peripheral component interconnect (PCI) express interface.

[0047] In the illustrated example, at least one input device 822 is connected to the interface circuit 820. The input device(s) 822 permit(s) a user to enter data and commands into the processor 812. The input device(s) can be implemented by, for example, an audio sensor, a microphone, a camera (still or video), a keyboard, a button, a mouse, a touchscreen, a track-pad, a trackball, and/or a voice recognition system.

[0048] At least one output device 824 is also connected to the interface circuit 820 of the illustrated example. The output device(s) 824 can be implemented, for example, by display devices (e.g., a light emitting diode (LED), an organic light emitting diode (OLED), a liquid crystal display, a cathode ray tube display (CRT), a touchscreen, a tactile output device, a light emitting diode (LED), a printer and/or speakers). The interface circuit 820 of the illustrated example, thus, may include a graphics driver card, a graphics driver chip or a graphics driver processor.

[0049] The interface circuit 820 of the illustrated example also includes a communication device such as a transmitter, a receiver, a transceiver, a modem and/or network interface card to facilitate exchange of data with external machines (e.g., computing devices of any kind) via a network 826 (e.g., an Ethernet connection, a digital subscriber line (DSL), a telephone line, coaxial cable, a cellular telephone system, etc.).

[0050] The processor platform 800 of the illustrated example also includes at least one mass storage device 828 for storing executable instructions (e.g., software) and/or data. Examples of such mass storage device(s) 828 include floppy disk drives, hard drive disks, compact disk drives, Blu-ray disk drives, RAID systems, and digital versatile disk (DVD) drives.

[0051] The coded instructions 832 of FIGS. 3, 4, 5, 6, and/or 7 may be stored in the mass storage device 828, in the local memory 813 in the volatile memory 814, in the non-volatile memory 816, and/or on a removable tangible machine readable storage medium such as a CD or DVD.

[0052] From the foregoing, it will be appreciated that the above disclosed methods, apparatus and articles of manufacture provide for adjustable hybrid of synchronous graph processing and asynchronous processing. In examples

herein, a set of nodes of a graph may be selected using adjustable (via a user interface) hybrid graph processing settings (e.g., using characteristics of nodes or priorities similar to asynchronous graph processing) and the selected set of nodes is processed in parallel (similar to synchronous graph processing). Accordingly, examples herein provide for the advantages of synchronous processing (the ability to process nodes in parallel) as well the advantages of asynchronous processing (the ability to process nodes in order of importance).

[0053] Although certain example methods, apparatus and articles of manufacture have been disclosed herein, the scope of coverage of this patent is not limited thereto. On the contrary, this patent covers all methods, apparatus and articles of manufacture fairly falling within the scope of the claims of this patent.

CLAIMS

What is claimed is:

1. A method comprising:

identifying characteristics associated with nodes of a graph;

selecting a set of nodes from the nodes to be processed based on hybrid graph processing settings and the characteristics; and

executing, via a processor, the selected set of nodes in parallel to process the graph.

2. The method as defined in claim 1, further comprising:

identifying priorities of the nodes from the characteristics;

identifying a threshold priority from the hybrid graph processing settings;

determining that the priorities of the nodes in the set of the nodes satisfy the threshold priority; and

selecting the set of nodes based on the priorities of the nodes in the set of nodes satisfying the threshold priority.

3. The method as defined in claim 1, further comprising:

sorting the nodes based on priorities in the characteristics of the nodes;

determining a threshold number of nodes to be selected, the threshold number corresponding to a magnitude of the set of nodes; and

selecting the threshold number of nodes from the sorted nodes with the highest priorities.

4. The method as defined in claim 1, further comprising:

assigning each of the nodes of the graph to a bin of a plurality of bins based on priorities associated with the nodes, the characteristics comprising the priorities;

selecting the set of nodes from a number of bins based on the hybrid graph processing settings.

5. The method as defined in claim 4, determining the number of bins from the hybrid graph processing settings.

6. The method as defined in claim 4, wherein the number of bins corresponds to a number of bins including nodes having a priority that satisfies a threshold priority.

7. The method as defined in claim 1, further comprising updating the characteristics of the selected set of nodes after executing the selected set of nodes.

8. The method as defined in claim 7, further comprising updating the characteristics of the nodes by adjusting a priority for each of the selected nodes based on at least one of execution time, execution urgency, or node type.

9. An apparatus for processing a graph:

a graph receiver to buffer a graph for processing, the graph comprising a plurality of nodes;

a node selector to select a set of nodes from the plurality of nodes based on priorities of the nodes and hybrid graph processing settings;

a settings manager to provide the graph processing settings to the node selector to select the nodes; and

a hybrid execution engine to execute the set of nodes in parallel in a first iteration to process the graph and update the priorities of the nodes based on the execution, the updated priorities to be used for a second subsequent iteration of processing the graph.

10. The apparatus of claim 9, wherein the node selector is to sort the plurality of nodes based on the priority for the first iteration.

11. The apparatus of claim 10, wherein the node selector is to re-sort the nodes after the hybrid execution engine executes the set of nodes in parallel in the first iteration.

12. A non-transitory machine readable medium comprising instructions that, when executed, cause a machine to at least:

process a first set of nodes of a graph in parallel in a first iteration of processing a graph, the first set of nodes selected from nodes of the graph based on priorities of the nodes; and

determine updated priorities for each of the first set of nodes based on the processed first set of nodes; and

provide the updated priorities for each of the first set of nodes for a second iteration subsequent to the first iteration.

13. The non-transitory machine readable medium of claim 12, wherein the instructions further cause the machine to:

sort the nodes of the graph prior to the processing the first set of nodes in the first iteration; and

re-sort the nodes of the graph after the first iteration and before the second iteration.

14. The non-transitory machine readable medium of claim 12, wherein the instructions further cause the machine to:

determine the updated priorities for each of the first set of nodes by determining an execution status of the nodes indicating execution time remaining until convergence of the node.

15. The non-transitory machine readable medium of claim 12, wherein the instructions further cause the machine to:

select a second set of nodes based on the updated priorities, and

process the second set of nodes in the second iteration.

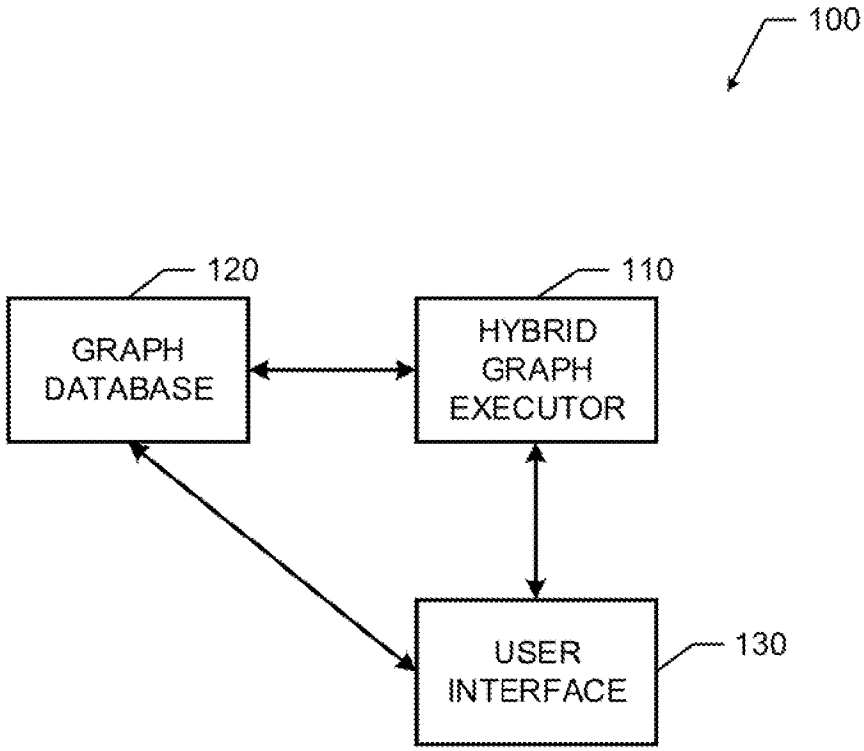
100
120
GRAPH DATABASE
110
HYBRID GRAPH EXECUTOR
USER INTERFACE
130

FIG. 1

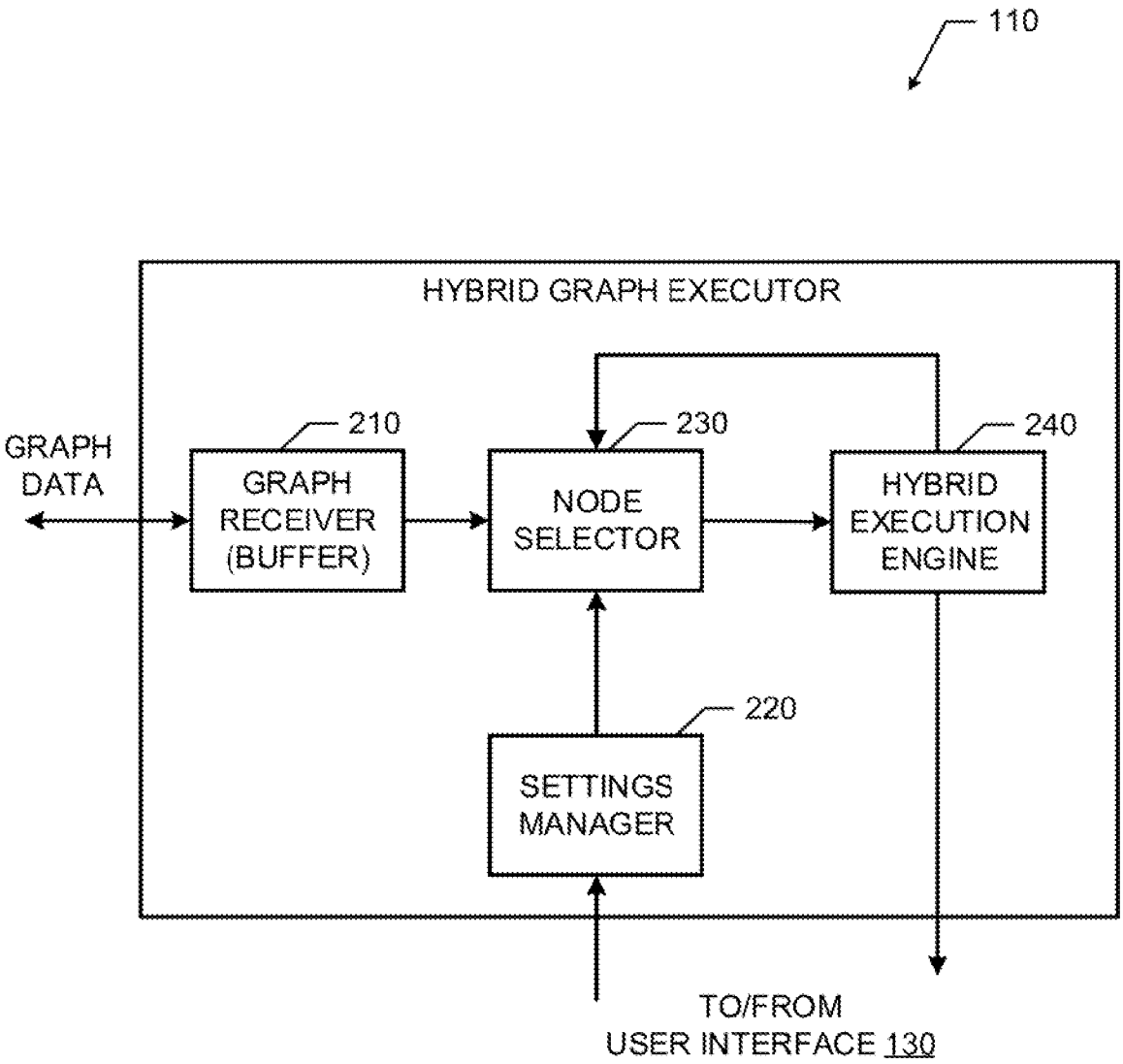
110
HYBRID GRAPH EXECUTOR
GRAPH DATA
210
GRAPH RECEIVER (BUFFER)
230
NODE SELECTOR
240
HYBRID EXECUTION ENGINE
220
SETTINGS MANAGER
TO/FROM USER INTERFACE 130

FIG. 2

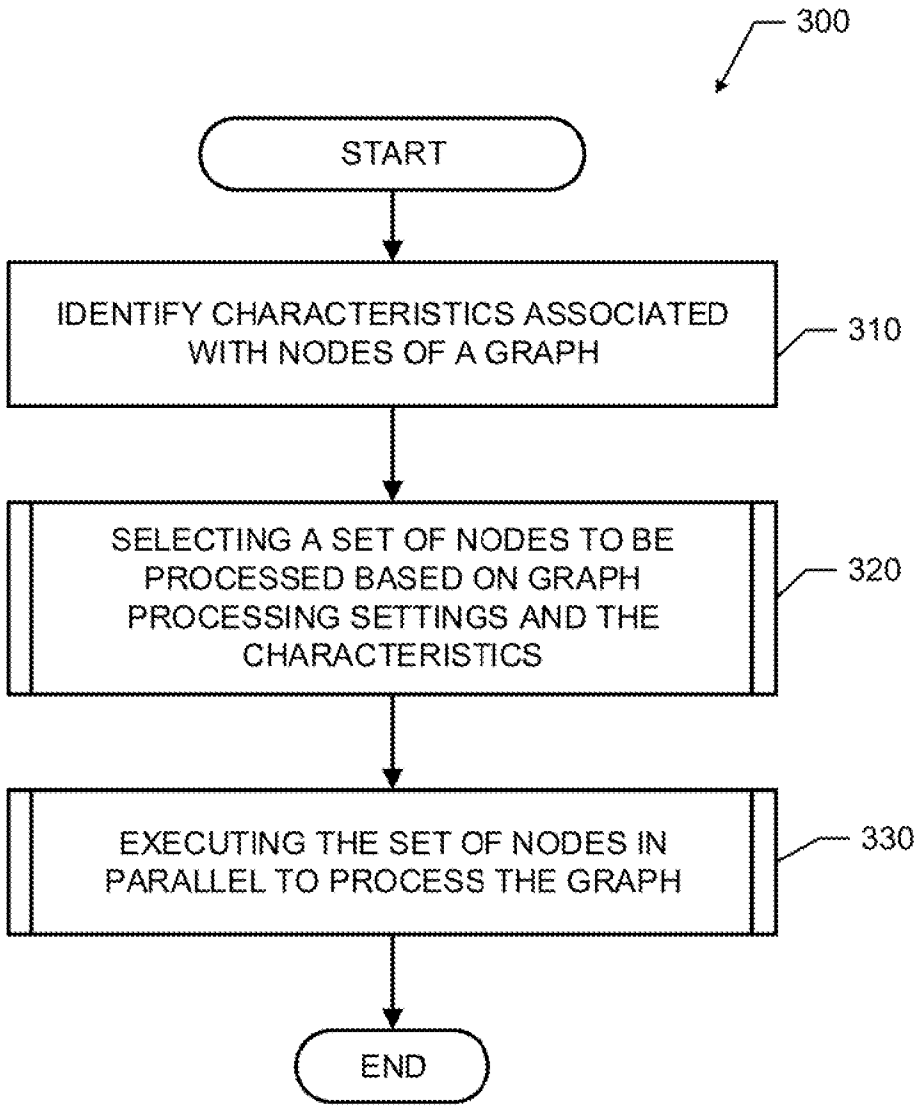
300
START
IDENTIFY CHARACTERISTICS ASSOCIATED WITH NODES OF A GRAPH
310
SELECTING A SET OF NODES TO BE PROCESSED BASED ON GRAPH PROCESSING SETTINGS AND THE CHARACTERISTICS
320
EXECUTING THE SET OF NODES IN PARALLEL TO PROCESS THE GRAPH
330
END

FIG. 3

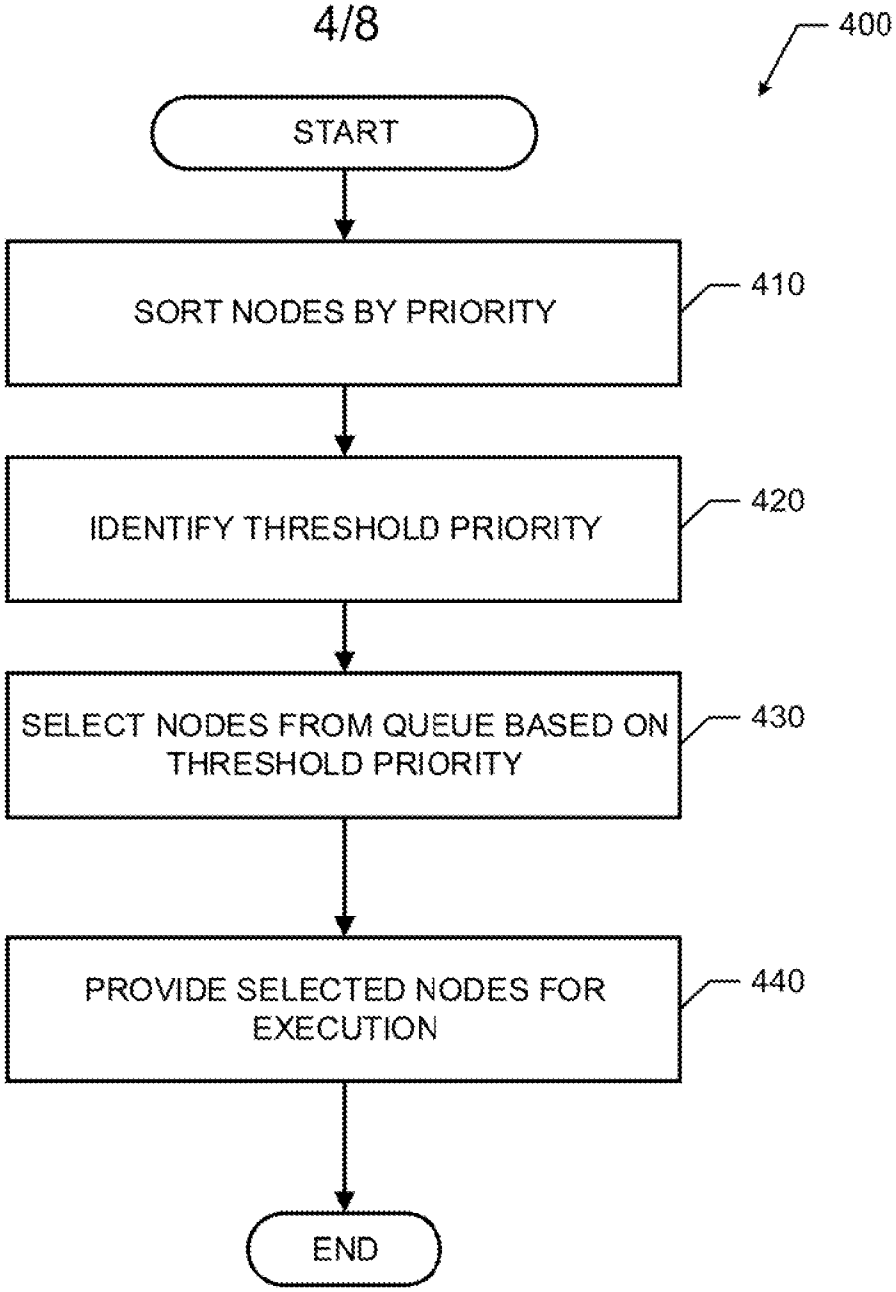
400
START
SORT NODES BY PRIORITY
410
IDENTIFY THRESHOLD PRIORITY
420
SELECT NODES FROM QUEUE BASED ON THRESHOLD PRIORITY
430
PROVIDE SELECTED NODES FOR EXECUTION
440
END

FIG. 4

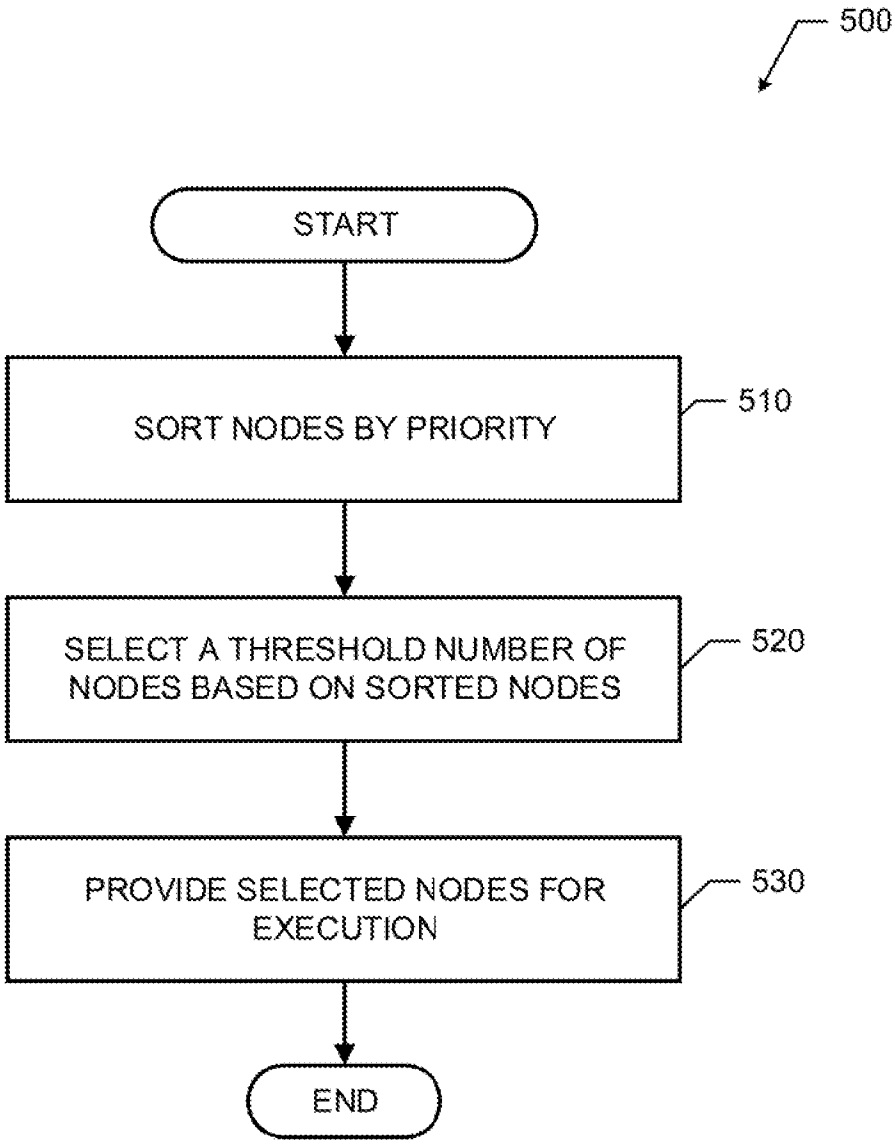
500
START
SORT NODES BY PRIORITY
510
SELECT A THRESHOLD NUMBER OF NODES BASED ON SORTED NODES
520
PROVIDE SELECTED NODES FOR EXECUTION
530
END

FIG. 5

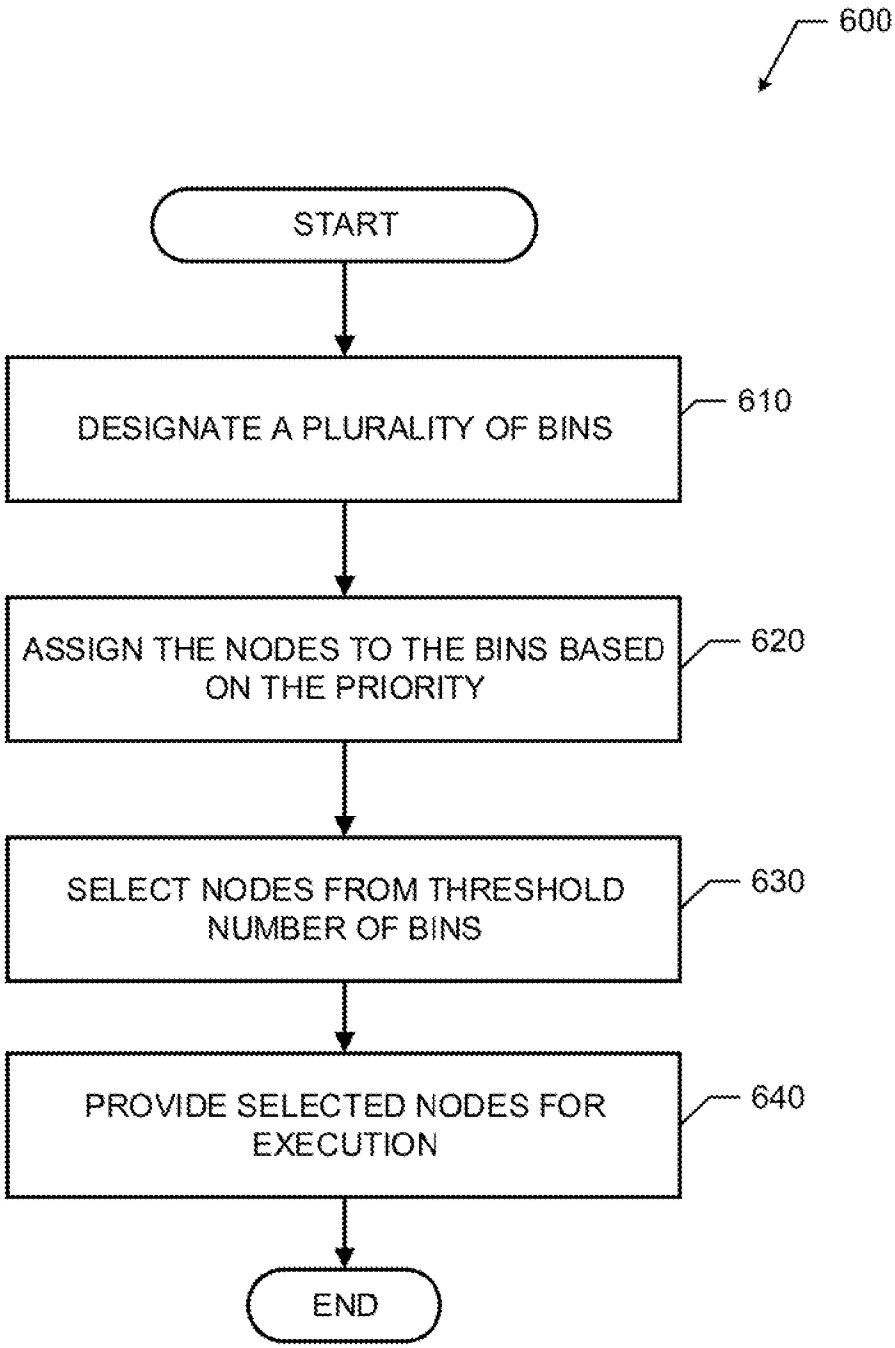
600
START
DESIGNATE A PLURALITY OF BINS
610
ASSIGN THE NODES TO THE BINS BASED ON THE PRIORITY
620
SELECT NODES FROM THRESHOLD NUMBER OF BINS
630
PROVIDE SELECTED NODES FOR EXECUTION
640
END

FIG. 6

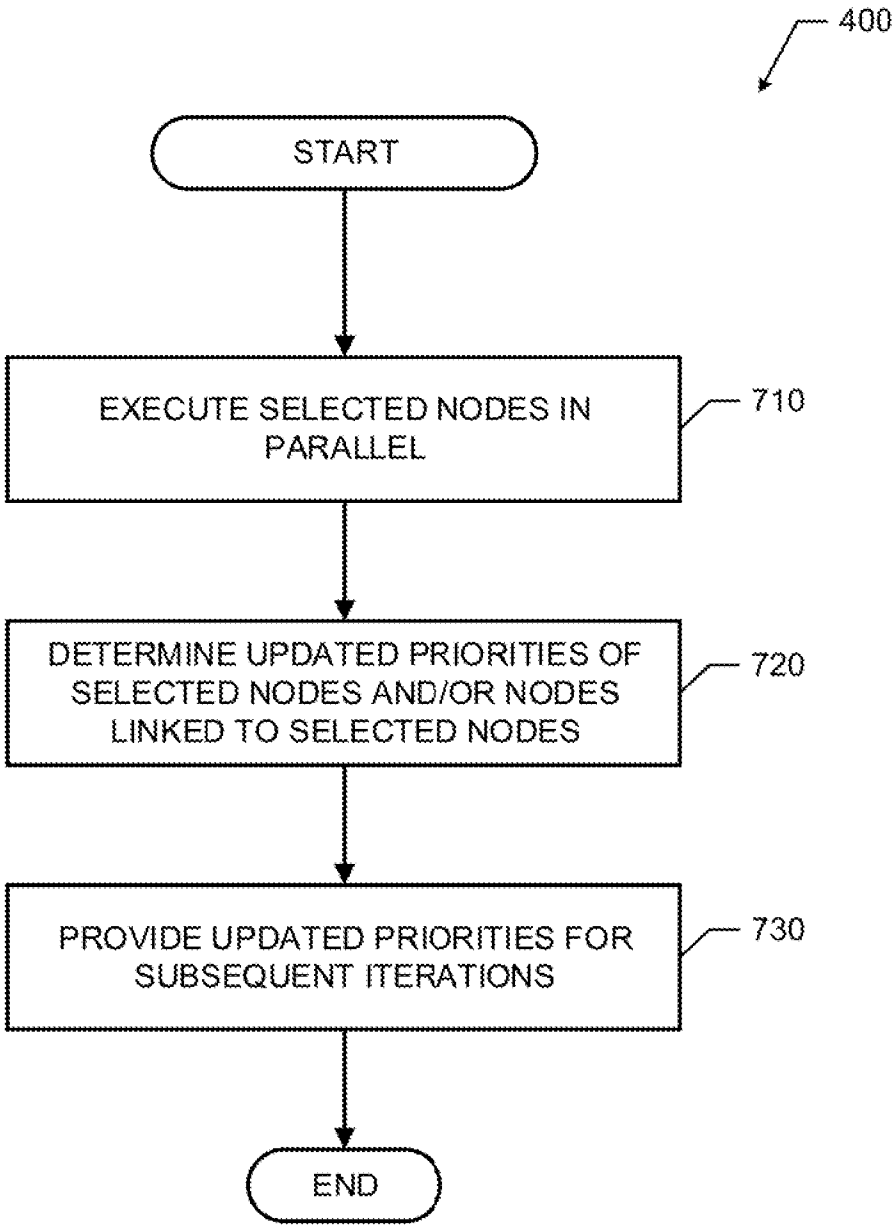
400
START
EXECUTE SELECTED NODES IN PARALLEL
710
DETERMINE UPDATED PRIORITIES OF SELECTED NODES AND/OR NODES LINKED TO SELECTED NODES
720
PROVIDE UPDATED PRIORITIES FOR SUBSEQUENT ITERATIONS
730
END

FIG. 7

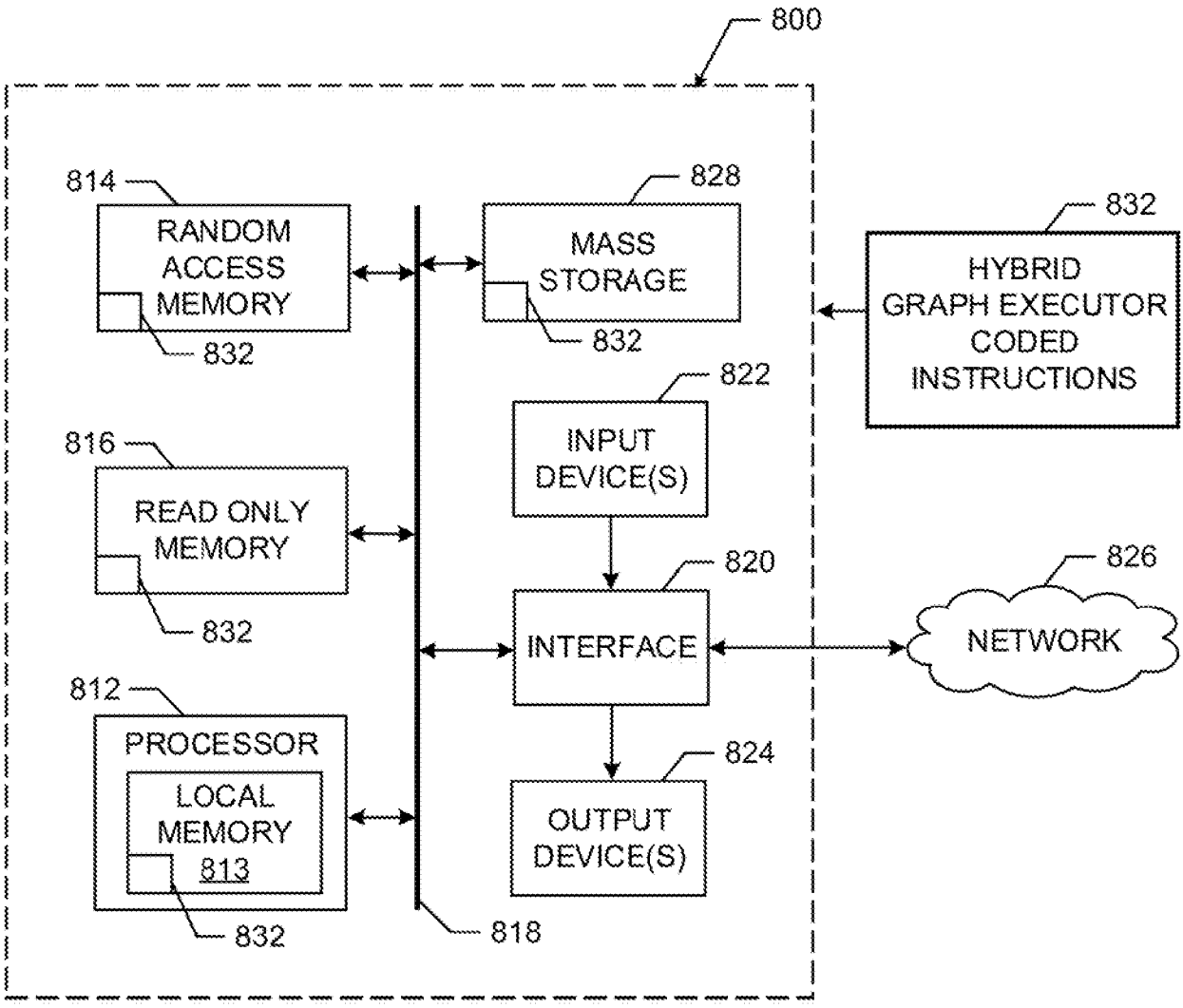
800
814
RANDOM ACCESS MEMORY
832
828
MASS STORAGE
832
832
HYBRID GRAPH EXECUTOR CODED INSTRUCTIONS
822
INPUT DEVICE(S)
816
READ ONLY MEMORY
832
820
826
NETWORK
INTERFACE
812
PROCESSOR
LOCAL MEMORY 813
832
824
OUTPUT DEVICE(S)
818

FIG. 8

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2015/048728

A. CLASSIFICATION OF SUBJECT MATTER

G06F 17/30(2006.01)i, G06F 17/00(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F 17/30; G06F 9/48; H04L 29/08; G06F 9/50; G06F 17/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords: graph processing, hybrid, priority, setting, parallel, and similar terms.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2014-0136553 A1 (INTERNATIONAL BUSINESS MACHINES CORP.) 15 May 2014 See paragraphs [0070]-[0075]; claim 1; and figures 5A-5C.	1-15
A	WENLEI XIE et al., "Fast Iterative Graph Computation with Block Updates," In: Proceedings of the VLDB Endowment, Vol. 6, No. 14, pp. 2014-2025, September 2013 (http://dl.acm.org/citation.cfm?id=2556581) See pages 2015 and 2024.	1-15
A	US 2015-0006606 A1 (GOOGLE INC.) 01 January 2015 See paragraphs [0036]-[0047] and figure 4.	1-15
A	US 2014-0380322 A1 (SAP AG) 25 December 2014 See paragraphs [0027]-[0030] and [0049]-[0054]; claims 1-2; and figures 1 and 5.	1-15
A	US 2015-0067695 A1 (MASAKI HAMAMOTO et al.) 05 March 2015 See paragraphs [0045]-[0046] and [0065]-[0075]; and figures 2 and 10-11.	1-15

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents,such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
31 May 2016 (31.05.2016)	**31 May 2016 (31.05.2016)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea
Facsimile No. +82-42-481-8578

Authorized officer
NHO, Ji Myong
Telephone No. +82-42-481-8528

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2015/048728

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2014-0136553 A1	15/05/2014	US 2014-0136555 A1	15/05/2014
US 2015-0006606 A1	01/01/2015	EP 3014446 A1	04/05/2016
		WO 2014-210501 A1	31/12/2014
US 2014-0380322 A1	25/12/2014	EP 2819009 A2	31/12/2014
		EP 2819009 A3	06/05/2015
		US 9329899 B2	03/05/2016
US 2015-0067695 A1	05/03/2015	WO 2013-145001 A1	03/10/2013