US 20180121564A1

(19) United States
(12) Patent Application Publication
Li et al.

(10) Pub. No.: US 2018/0121564 A1
(43) Pub. Date: May 3, 2018

(54) HIGH-SPEED GRAPH PROCESSOR

(71) Applicant: **Wisconsin Alumni Research Foundation**, Madison, WI (US)

(72) Inventors: **Jing Li**, Madison, WI (US); **Jialiang Zhang**, Middleton, WI (US); **Soroosh Khoram**, Madison, WI (US)

(21) Appl. No.: **15/340,340**

(22) Filed: **Nov. 1, 2016**

Publication Classification

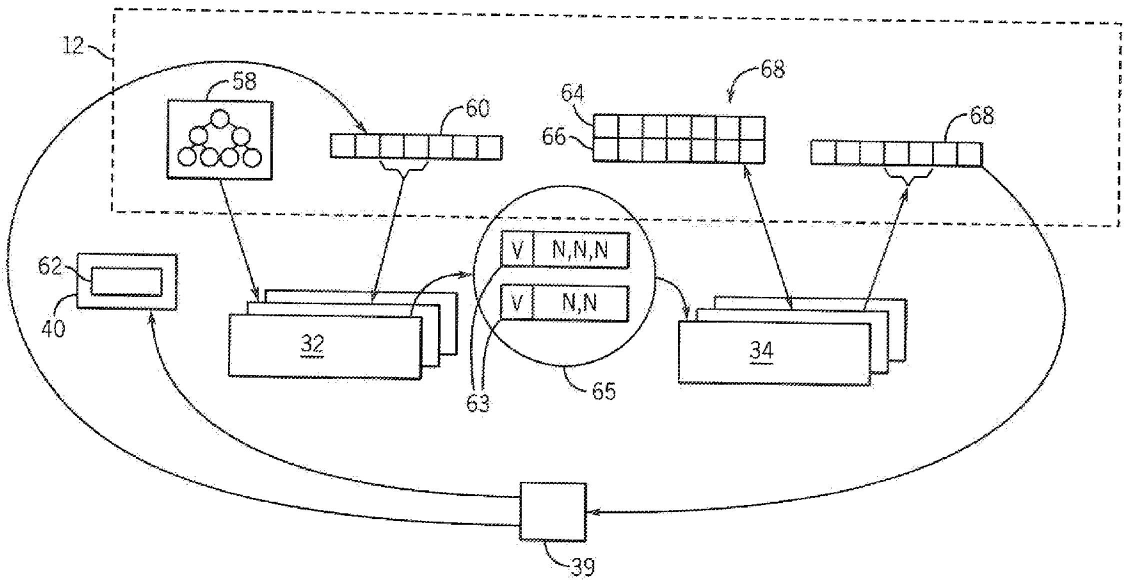
(51) **Int. Cl.**

G06F 17/30	(2006.01)
G11C 5/06	(2006.01)
G11C 7/00	(2006.01)

(52) **U.S. Cl.**
CPC .. ***G06F 17/30958*** (2013.01); ***G06F 17/30979*** (2013.01); ***G11C 7/00*** (2013.01); ***G11C 5/06*** (2013.01)

(57) **ABSTRACT**

A computer architecture for graph processing employs a high-bandwidth memory closely coupled to independent processing elements for searching through a graph using a first set of processing elements operating simultaneously to determine neighbors to a current frontier and second processing elements operating simultaneously to determine a next frontier, this process being repeated to search through graph nodes.



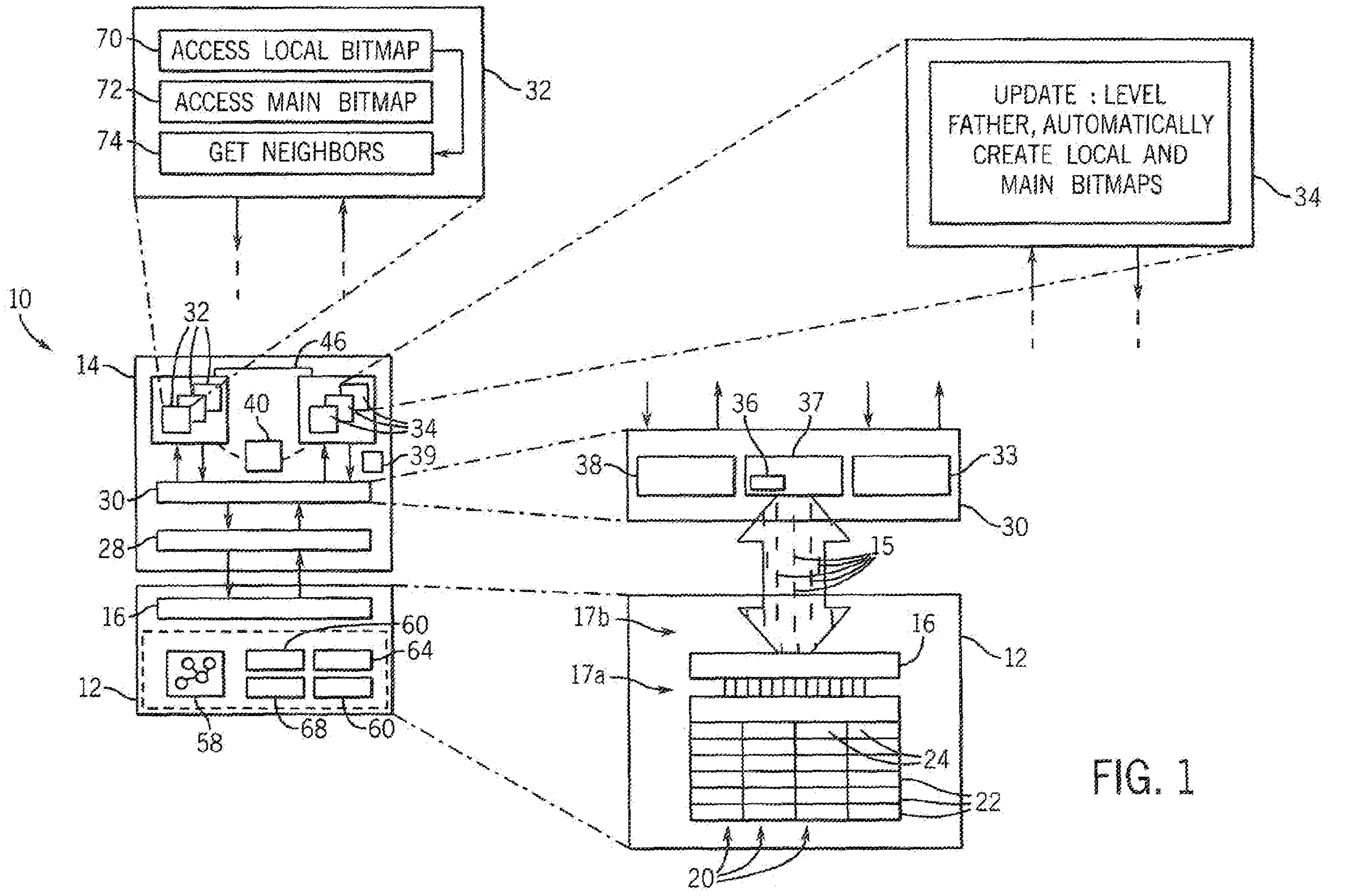

ACCESS LOCAL BITMAP
ACCESS MAIN BITMAP
GET NEIGHBORS
UPDATE : LEVEL FATHER, AUTOMATICALLY CREATE LOCAL AND MAIN BITMAPS
10
12
14
15
16
17a
17b
20
22
24
28
30
32
33
34
36
37
38
39
40
46
58
60
64
68
70
72
74

FIG. 1

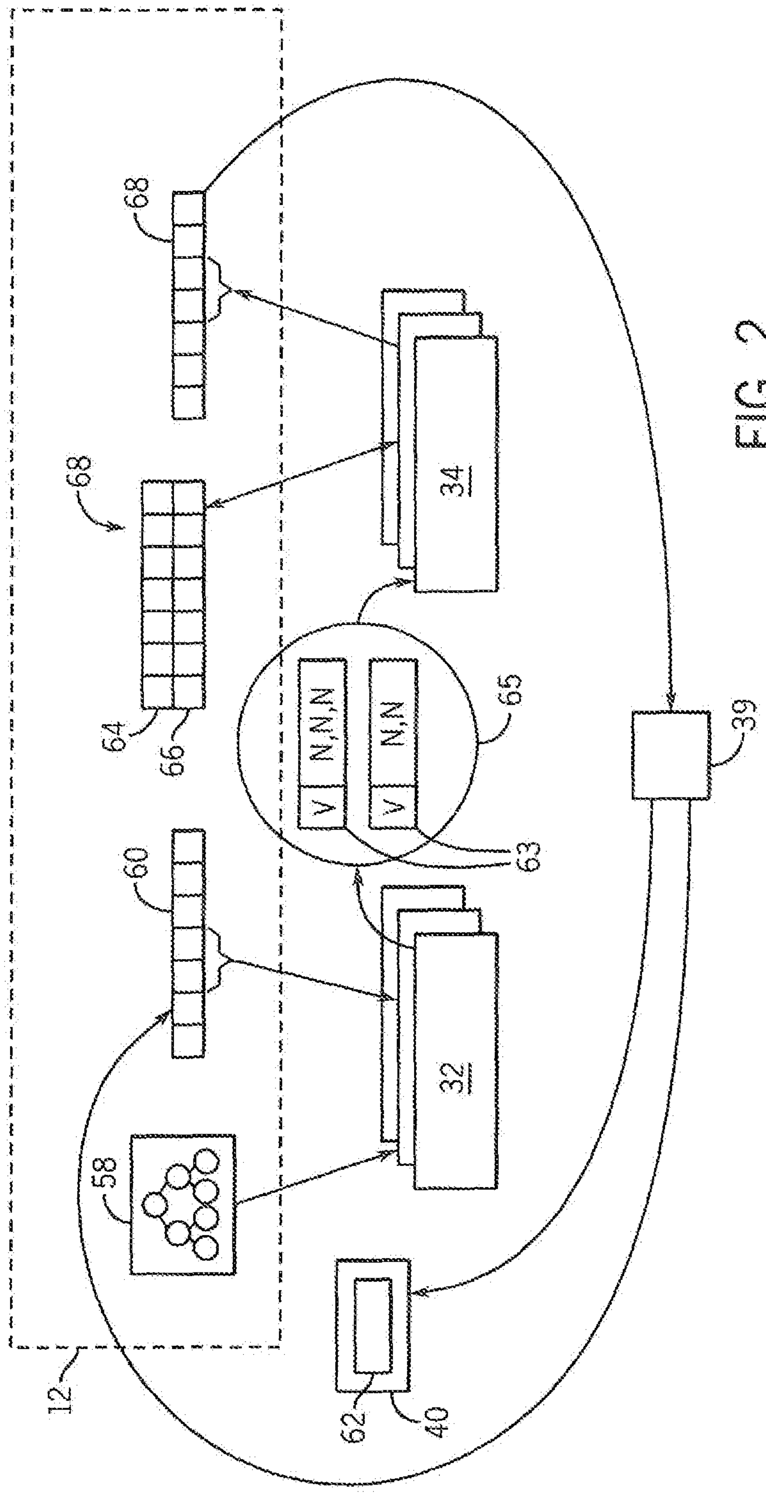


FIG. 2

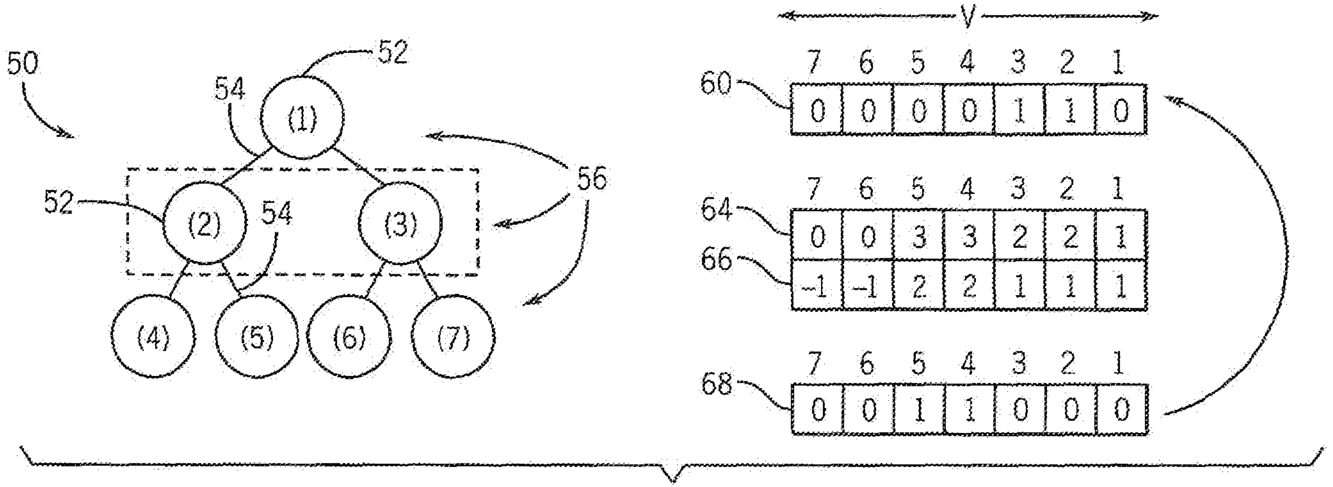

50
52
54
(1)
52
(2)
54
(3)
56
(4)
(5)
(6)
(7)
V
7 6 5 4 3 2 1
60
0 0 0 0 1 1 0
64
0 0 3 3 2 2 1
66
-1 -1 2 2 1 1 1
68
0 0 1 1 0 0 0

FIG. 3

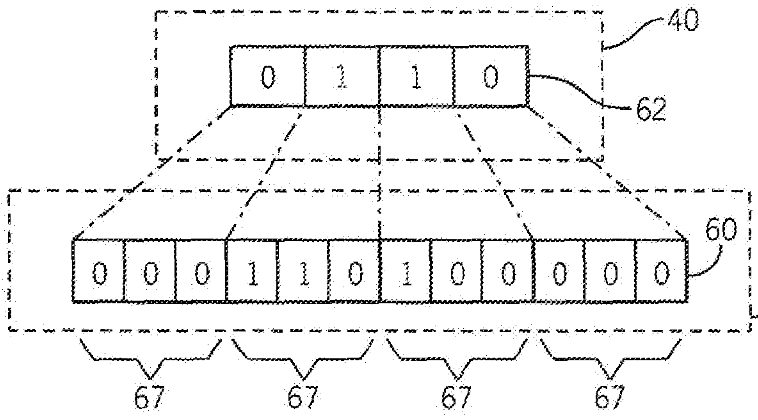

40
0 1 1 0
62
60
0 0 0 1 1 0 1 0 0 0 0 0
12
67
67
67
67

FIG. 4

HIGH-SPEED GRAPH PROCESSOR

STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

[0001] - -

CROSS REFERENCE TO RELATED APPLICATION

[0002] - -

BACKGROUND OF THE INVENTION

[0003] The present invention relates to a computer architecture, and in particular, to a computer architecture suitable for problems such as the processing of large, sparse graphs.

[0004] A graph is a data structure describing a set of vertices (nodes) interconnected by edges to form a network. Graphs provide a useful model of many real-world structures, for example, relationships between webpages (where the vertices describe webpages and the edges links between webpages) or individuals in a social network (where the vertices describe individuals and the edges describe their friends or interests). Graphs are also used in the fields of neural networks and bioinformatics.

[0005] The processing of graphs may be used to answer questions such as: what are the closest connections between two vertices, or what vertices share a particular set of common connections, and the like. Such processing may employ a "breadth first" or similar search to explore the graph moving among vertices according to their edge connections.

[0006] Graph data structures can quickly become large making graph processing slow. Currently, the social networking site of Facebook is described by a graph with over 1 billion vertices and more than 1 trillion edges.

SUMMARY OF THE INVENTION

[0007] The present invention provides a specialized processor suitable for processing graphs and similar data structures. In this regard, the inventors have recognized that graph processing speed is fundamentally limited by delays in transferring graph data between the memory and processor, a transfer that is exacerbated not only by the large size of the graphs but also by their sparseness which results in random and data dependent memory access patterns that prevent effective use of memory caches to speed data transfer.

[0008] For this reason, the present invention provides an architecture that closely couples a high-bandwidth, small access granularity memory (for example, a three-dimensional "memory cube") and a dedicated parallel processor, the latter providing parallel "mapping" and "reducing" that may operate on independent memory channels of the memory, greatly reducing the memory bottleneck for search operations on sparse graphs and the like.

[0009] Initial evaluation of this architecture suggests speed gains in the GRAPH500 benchmark of more than three times compared to standard computer architectures using graphs with a scale of 26 and edge factor of 16. Graph scale is the number of vertices in the graph and edge factor is the ratio between the total number of edges and the total number of vertices.

[0010] Specifically, the present invention provides a graph processor using a memory closely coupled to a set of independent processing circuits. The memory supports multiple independent memory channels for reading and writing memory data and is adapted to hold a graph data structure describing vertices connected by edges. The independent processing circuits communicate with the memory and program to search through the vertices of the graph data and include: (a) a first set of independent processing circuits operating to identify neighbors to a current frontier of the search by accessing the graph data structure; and (b) a second set of independent processing circuits operating to receive the identified neighbors and determine a new frontier by accessing the result data.

[0011] It is thus a feature of at least one embodiment of the invention to make use of high-bandwidth, fine granularity memory architectures, such as three-dimensional memory, to provide a processor well adapted to searches on sparse graphs by using multiple processing circuits associated with the multiple memory channels that may operate in parallel using the different small data accesses.

[0012] Multiple of the first set of independent processing circuits may identify neighbors of different portions of the current frontier simultaneously and multiple of the second independent processing circuits may identify portions of the new frontier simultaneously.

[0013] It is thus a feature of at least one embodiment of the invention to provide a graph searching process into components that can be made independent with respect to memory access to fully exploit the bandwidth of the memory.

[0014] The data accessed in the memory by the first set of independent processing circuits may be disjoint in memory from the data accessed by the memory by the second set of processing circuits for simultaneous access.

[0015] It is thus a feature of at least one embodiment of the invention to prevent memory access conflicts between the first and second set of independent processing circuits, for example, as distinct from memory dependence conflicts resulting from one process needing to wait for data produced by another process.

[0016] The first and second sets of independent processing circuits may operate on frontiers of a single level of vertices at a time, where a level is determined by edge distance from a source vertex.

[0017] It is thus a feature of at least one embodiment of the invention to provide a level "synchronize" process eliminating the complexity of out-of-order processing to provide a simple architecture.

[0018] At the conclusion of the calculation of the current frontier and new frontier for a level, the new frontier level determined by the second set of independent processing circuits may become the current frontier used by the first set of independent processing circuits for a succeeding level.

[0019] It is thus a feature of at least one embodiment of the invention to provide a recursive search approach simplifying architecture definition.

[0020] The second set of independent processing circuits may further write to the result data acquired during the search to the memory.

[0021] It is thus a feature of at least one embodiment of the invention to provide an architecture that can perform arbitrary data collection during a searching process using the second set of independent processors.

[0022] The result data may be distance for each vertex from a source vertex at which the search started and a father vertex for each vertex.

[0023] It is thus a feature of at least one embodiment of the invention to provide a common operation on graphs of determining edge-based distances between nodes.

[0024] The memory further holds a bit vector describing the current frontier and a bit vector describing a next frontier in the search and wherein the first set of independent processing circuits reads the current frontier bit vector to identify vertices in the frontier, and the second set of independent processing circuits writes to the new frontier bit vector based on the identified neighbors that have not been searched.

[0025] It is thus a feature of at least one embodiment of the invention to provide simple data structures representing search frontiers that can be accessed in portions using the fine granularity of the memory facilitating parallel operation. Ideally the bit vector is greater than the smallest access page of the memory times the number of independent processor circuits in the first or second set. In this way each of the processors can be fully exploited in parallel for processing the bit vector.

[0026] The second set of independent processing circuits may write result data associated with each vertex in the next frontier and may determine which of the identified neighbors have not been searched using the result data for those vertices.

[0027] It is thus a feature of at least one embodiment of the invention to allow the result data to both record results and to separate identified neighbors determined by the first set of independent processors from the next frontier by eliminating identified neighbors that already have results.

[0028] The graph processor may provide a local memory smaller in size than the high-bandwidth memory holding a compressed bitmap smaller than the current frontier bitmap having bits representing compressed form of the frontier bitmap and wherein the first set of independent processing circuits identifies neighbors to a current frontier of the search by first accessing the compressed bitmap and then accessing the frontier bitmap only if the compressed bitmap indicates a set bit will exist in the predetermined range of the frontier bitmap.

[0029] It is thus a feature of at least one embodiment of the invention to reduce the need to consume memory bandwidth by examining a small local memory to determine whether a bitmap in memory needs to be accessed.

[0030] The bits of the compressed bitmap may represent the logical OR of different predetermined ranges of the frontier bitmap.

[0031] It is thus a feature of at least one embodiment of the invention to provide a simple lossy compression system that positively indicates when the bitmap in memory need not be accessed.

[0032] The second set of independent processing circuits operates to atomically update the new frontier bit vector and the compressed bit vector.

[0033] It is thus a feature of at least one embodiment of the invention to provide a highly parallel operating environment that can use the memory to resolve data access conflicts.

[0034] The channels of the memory may communicate serial packets tagged with respect to the independent processing circuit making the access request.

[0035] It is thus a feature of at least one embodiment of the invention to provide a mechanism for high-speed asynchronous access to the memory reducing queuing waits.

[0036] The number of independent processing circuits may be no less than the number of independent memory channels.

[0037] It is thus a feature of at least one embodiment of the invention to fully utilize the multiple channels of memory.

[0038] The memory may be a three-dimensional memory employing through-silicone-vias.

[0039] It is thus a feature of at least one embodiment of the invention to provide a memory that may be closely integrated into the architecture of the present invention by employing a highly dense memory array.

[0040] These particular objects and advantages may apply to only some embodiments falling within the claims and thus do not define the scope of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[0041] FIG. **1** is a block diagram of the architecture of the present invention showing a processing unit coupled to a high-bandwidth memory and indicating in several exploded details, the independent mapper elements and reducer elements and an interconnect operating between the mapper elements and reducer elements and the high-bandwidth memory;

[0042] FIG. **2** is a dataflow representation of a single mapper element and reducer element showing the disjoint data used by each element permitting access-conflict-free parallel operation;

[0043] FIG. **3** is a logical representation of a simplified graph showing memory structures of a current frontier bitmap, a next frontier bitmap and a result array recording level and father information showing operation of the mapper elements and reducer elements implemented by the present architecture; and

[0044] FIG. **4** is a logical representation of a current frontier bitmap implemented with one layer of indirection for reduced memory access bandwidth consumption.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

[0045] Referring now to FIG. **1**, a graph processor **10** of the present invention may provide for a high-bandwidth memory **12** closely coupled to a processor unit **14**, for example, incorporated into the same integrated circuit or adjacent integrated circuits for high-speed communication.

[0046] The high-bandwidth memory **12** may provide data access through multiple independent lanes **15** (being logical channels that may differ from actual physical connections) using full-duplex packetized communication. Each packet can include a tag (for example, identifying the source of the memory access request and the destination for the accessed data) as well as a desired address of the data being accessed and may allow several different access granularities. In one embodiment the granularity may range from 16 bytes to 128 bytes of data. This ability to provide for adjustable and low granularity allows optimizing the data transfer to prevent the transfer of data that is not required for a calculation but that would otherwise be counted against the total bandwidth available between processor unit **14** and memory **12**.

[0047] The independent links **15** may be implemented by a group of serializer/deserializer circuit (links) **16** communicating parallel data **17***a* with storage elements of the memory **12** and translating the parallel data **17***a* to or from serial packet data **17***b* providing communication with the processor unit **14**.

[0048] The memory **12** may be organized into multiple vaults **20** that can be simultaneously and independently accessed. The vaults **20** are divided into multiple banks **22** providing memory cells connected using vertical (through-silicone-vias) TSV technology and are each managed by a vault controller **24**. Addressing the memory **12** employs an address consisting of a vault address, a bank address, a random access row and column address (within a bank) and a byte address. The granularity of data exchange may be 16 bytes, 32 bytes, 64 bytes or 128 bytes controlled by the size of the bite address. Generally, each packet may provide data from only one vault **20**.

[0049] In one embodiment, the high-bandwidth memory **12** may be a Hybrid Memory Cube (HMC) memory constructed according to the specifications of the Hybrid Memory Cube Consortium being a joint collaboration between Micron and Samsung working with a variety of other manufacturers with components commercially available from Micron under the tradename HMC Gen2 (HMC-15 G-SR). This device generally provides extremely high bandwidth (amount of data transfer per second) providing two links (memory channels) of full duplex communication each consisting of eight 15 gigabits per second lanes, and providing a total bandwidth of 30 gigabytes per second. Other HMC components can support up to 4 links, each with sixteen 30 gigabits per second lanes, and a aggregated bandwidth of 240 gigabytes per second.

[0050] The processor unit **14** will generally be a monolithic integrated circuit to provide for high-speed interconnection between its elements with short run lines but may be any similar structures including hybrid integrated circuits comprised of multiple interconnected integrated circuits assembled using integrated circuit technologies. The processor unit **14** is preferably a dedicated integrated circuit (for example, implemented using an FPGA) rather than the general-purpose computer processor.

[0051] The processor unit **14** provides two pools of independent processing circuits. The first pool comprises a set of independent mapper elements **32** and the second pool a set of independently reducer elements **34**, each which will be described in greater detail below. In this context, independence generally indicates that each of these mapper elements **32** and reducer elements **34** may execute in parallel. Generally, the number of mapper elements **32** and reducer elements **34** will be no less than the number of independent lanes **15** of the memory **12** to maximize the potential for data transmission.

[0052] The memory **12** connects to the processor unit **14** using a shared memory controller **28** that converts the high-speed data from the lanes **15** to and from the memory **12** into parallel lower speed data compatible with the processor unit **14**. This shared memory controller **28** in turn communicates with interconnect circuit **30** which includes buffers and serves to dynamically match data received from the memory **12** in serial packet data **17***b* associated with previous corresponding data requests from various mapper elements **32** and reducer elements **34**, to the appropriate requesting mapper element **32** and reducer element **34**.

[0053] In this regard, the interconnect circuit **30** may provide for a downstream queue **33** receiving write requests or read requests from the mapper elements **32** and reducer elements **34** which are then provided to a command buffer **37**. Each of these requests includes a tag identifying the particular mapper element **32** or reducer element **34** originating the request. The command buffer **37** logs the tags associated with those requests in a tag memory **36** and the downstream queue **33** buffers those requests as necessary.

[0054] Data received from the memory controller **28** by an upstream queue **38** of interconnect circuit **30** (which may also buffer this data) is then passed to the command buffer **37** for matching to a tag in the received data packet using the tags of tag memory **36**. This matching allows the interconnect circuit **30** to forward this received data to the requesting mapper element **32** or reducer element **34**. As data is received from the memory **12**, the tags in tag memory **36** are retired so that the tag memory **36** may be small and buffers are emptied. In this respect, the command buffer **37** of the interconnect circuit **30** serves as a router to ensure that data is correctly provided between individual mapper elements **32** and reducer elements **34** and memory **12** while allowing flexible transmission using a packet system.

[0055] As well as communicating data through the interconnect circuit **30**, each of the mapper elements **32** and reducer elements **34** may also communicate directly with each other over a dedicated internal on-chip bus **46** that will be described further below. In addition, each of the mapper elements **32** may read from a local memory **40** as will be described below. This local memory **40** is on the same integrated circuit as the mapper elements **32** and reducer elements **34** for high-speed communication and serves to minimize unnecessary communication with the main memory **12**.

[0056] The processor unit **14** may also include sequencing logic **39** that will serve to advance the searching process in a synchronized fashion through levels as will be described.

[0057] Referring now to FIGS. **1**-**3**, the mapper elements **32** and reducer elements **34** may be figured to process a graph **50** logically represented as a set of vertices **52** (in this example labeled (1)-(7) of number V) connected by edges **54**. A search may be conducted of the graph **50** starting with a search vertex (labeled (1)) and proceed through various levels **56** determined by the number of edges **54** separating the vertices **52** of the graph **50** from the source vertex (1). The graph **50** may be represented in the memory **12** by a graph description **58**, for example, providing a listing of graph vertices **52** and their associated edges **54**. In one nonlimiting example, the graph description **58** may be in the form of an adjacency list which provides pointers associated with each vertex **52** that point to different segments of an adjacency array describing each neighbor vertex of vertex **52** connected by an edge **54**. Other representations are also contemplated.

[0058] In conducting a search of the graph **50**, the mapper elements **32** rely on a current frontier bitmap **60** stored in memory **12** providing a bit for each vertex **52** in the graph **50** that represents a frontier in expanding the search of the graph **50**. The current frontier bitmap **60** has a size equal to the number of vertices V in the graph **50**. Logical bits of the current frontier bitmap **60** are set if the vertex **52** is in the current search frontier representing the boundary between the vertices that have been searched and those that have not been searched.

[0059] Multiple mapper elements **32** may read the graph description **58** and the current frontier bitmap **60** in parallel (without data dependency conflicts) because the access is limited to reading. Generally too, each mapper element **32** will read only a portion of the current frontier bitmap **60** taking advantage of the small access granularity of the memory **12** to preserve memory bandwidth.

[0060] Using the graph description **58** and the current frontier bitmap **60**, each mapper element **32** will a create neighbor lists **63** associating a given vertex indicated by the current frontier bitmap **60** with all other vertices to which it is connected by means of edges **54**. As will be described below, these neighbor lists **63** are buffered in a pool **65** directly accessible to the reducer elements **34** over the on-chip bus **46** and thus do not require access to the memory **12** for writing.

[0061] The reducer elements **34** also communicate with the memory **12**, but with data structures that are disjoint from the data structures accessed by the mapper elements **32** so that there are no data access conflicts that would slow down the reading or writing of this data. In particular, the reducer elements **34** read the neighbor lists **63** and also may read and write a results vector **69** based on the search. As will be discussed below, the results vector **69** may be the desired results of the search of the graph **50**; however, the results vector **69** is also used in the process of identifying a next frontier bitmap **68** by the reducer elements **34** and thus provides two functions.

[0062] In one embodiment, the results vector **69** may be composed of level vector **64** and father vector **66** each providing an integer value for each vertex **52** in the graph **50**. The level vector **64** will store the level of the each particular vertex **52** in the graph **50** and the father vector **66** will store the father of the each particular vertex **52** as deduced during this searching process. Other results can also be accumulated during the searching process and the invention should not be considered limited to this particular result.

[0063] Multiple reducer elements **34** may read the neighbor lists **63** from the pool **65** and by reading the results vector **69** may determine which of the neighbors in neighbor lists **63** are new un-searched vertices **52** (and thus not having any results in the results vector **69**) and therefore are part of the next frontier bitmap **68** describing the expansion of the search. For this level, for each un-searched vertice **52**, a level (being the distance from the source vertex v_n and the un-searched vertex **52**) is determined and a father vertex **52** to the uncertain vertex **52** is determined to populate the level vector **64** and father vector **66** for those on searched vertices **52**.

[0064] Once the steps are complete, the next frontier bitmap **68** will then become the current frontier bitmap **60** and the sequencing logic **39** advances all mapper elements **32** and reducer elements **34** to the next level.

[0065] Referring now to FIGS. **1** and **3**, each mapper element **32** may execute in a pipeline a set of operations, the first of which is indicated by process blocks **70** and **72** to identify each vertex **52** in the level **56** of the current frontier using the current frontier bitmap **60**. Assuming the current level **56** being processed includes vertices (2) and (3), the current frontier bitmap **60** will show a set of zeros except for positions 2 and 3. While the mapper elements **32** may directly access the current frontier bitmap **60** for this purpose, the present invention contemplates that memory **12** bandwidth may be preserved by first consulting a compressed bitmap **62** held in local memory **40** as indicated by pipeline block **70**.

[0066] As shown in FIG. **4**, each bit of the compressed bitmap **62** may map multibit portions **67** of the current frontier bitmap **60** stored in memory **12** by performing a logical OR of the bits in each portion **67** as indicated by pipeline block **70**. Thus, for example, a three-bit portion of bit vectors **60** having values 000 will map to a single value of 0 in the corresponding bit of the compressed bitmap **62**. Similarly a three-bit portion of the current frontier bitmap **60** having value 010 (or values 001, 011, 100, 110, or 111) will map to a single value of 1 in the corresponding bit of the compressed bitmap **62**.

[0067] By reviewing the compressed bitmap **62**, access to the current frontier bitmap **60** in memory **12**, per pipeline block **72**, may be completely avoided if the compressed bitmap **62** for the portions of current frontier bitmap **60** of interest, shows a value of zero. Otherwise, bitmap **60** in memory **12** is accessed at pipeline block **72** to determine exactly which vertices **52** are in the relevant portion of the current frontier bitmap **60**. In a sparse graph **50**, access to memory **12** may be frequently avoided.

[0068] At pipeline block **74**, for each bit that is set in the current frontier bitmap **60** representing a frontier vertex **52** on the current search frontier through the graph **50**, the graph description **58** is examined to identify all neighbor vertices to that given vertex **52**. So, for example, for vertex (2) in the current level, neighbor vertices (4) and (5) would be identified. This set of linked vertices (vertex (2) linked to vertices (4) and (5)) provides a neighbor list **63** that is forwarded to the reducer elements **34** through a queue pool **65** that allows any available reducer element **34** to take any available neighbor list **63**.

[0069] In contrast to the mapper elements **32**, the reducer elements **34** execute a single program step on each neighbor list **63**, looking at each neighbor vertex **52** to see if that vertex **52** has a corresponding entry (other than a null entry) in the results vector **69** (either father vector **66** or level vector **64**) indicating that this vertex **52** has already been reviewed in the searching process. For this purpose the results vector **69** will normally be initialized to null values. If there is other than a null entry for particular neighbor vertex **52**, the reducer element **34** writes to the next frontier bitmap **68** a value of one indicating that this neighbor vertex **52** is in the next frontier to be analyzed. The reducer element **34** also writes the level of that vertex **52** into the level vector **64** (being level 2 for vertices (4) and (5)) and the father vertex of those neighbor vertices **52** into the father vector **66** (being vertex (2) for vertices (4) and (5)) being the vertex associated with the neighbor in the neighbor list **63**. This process is repeated for each of the set bits of the current frontier bitmap **60**.

[0070] At the conclusion of the operation of the mapper elements **32** and reducer elements **34** for a given level, the level is incremented after the sequencing logic **39** and the current frontier bitmap **60** are replaced with values of the next frontier bitmap **68**. This can be done, for example, without access to the memory **12** by simply swapping address pointers. During this swapping, the mapper elements **32** and reducer elements **34** are restrained from further processing of the next level, for example, by flag bits or the like controlled by the sequencing logic **35**. At the same time the compressed bitmap **62** is updated. This updating may be done atomically with the updating of the next frontier bitmap **68** to avoid inconsistencies between the compressed bitmap **62** and the next frontier bitmap **68**.

[0071] The sequencing logic **39** moves through levels until the entire graph **50** is searched as may be determined by testing whether every value of results vector **69** has a non-null value.

[0072] The above process implements a "breadth first search" (BFS) to determine father nodes and levels for all vertices **52**. It will be appreciated that this same search strategy may be used for a variety of other graph operations and that this architecture is not limited to particular graph operations.

[0073] Certain terminology is used herein for purposes of reference only, and thus is not intended to be limiting. For example, terms such as "upper", "lower", "above", and "below" refer to directions in the drawings to which reference is made. Terms such as "front", "back", "rear", "bottom" and "side", describe the orientation of portions of the component within a consistent but arbitrary frame of reference which is made clear by reference to the text and the associated drawings describing the component under discussion. Such terminology may include the words specifically mentioned above, derivatives thereof, and words of similar import. Similarly, the terms "first", "second" and other such numerical terms referring to structures do not imply a sequence or order unless clearly indicated by the context.

[0074] When introducing elements or features of the present disclosure and the exemplary embodiments, the articles "a", "an", "the" and "said" are intended to mean that there are one or more of such elements or features. The terms "comprising", "including" and "having" are intended to be inclusive and mean that there may be additional elements or features other than those specifically noted. It is further to be understood that the method steps, processes, and operations described herein are not to be construed as necessarily requiring their performance in the particular order discussed or illustrated, unless specifically identified as an order of performance. It is also to be understood that additional or alternative steps may be employed.

[0075] References to an integrated circuit refer to monolithic integrated circuits or integrated circuits assembled from multiple dies for example using through silicon vias micro bumps or the like providing compact packages for rapid communication

[0076] It is specifically intended that the present invention not be limited to the embodiments and illustrations contained herein and the claims should be understood to include modified forms of those embodiments including portions of the embodiments and combinations of elements of different embodiments as come within the scope of the following claims. All of the publications described herein, including patents and non-patent publications are hereby incorporated herein by reference in their entireties.

What we claim is:

1. A graph processor comprising:

a memory supporting multiple independent memory channels for reading and writing memory data, the memory adapted to hold a graph data structure describing vertices connected by edges; and

a set of independent processing circuits communicating with the memory and program to search through the vertices of the graph data including:

(a) a first set of independent processing circuits operating to identify neighbors to a current frontier of the search by accessing the graph data structure; and

(b) a second set of independent processing circuits operating to receive the identified neighbors and determine a new frontier as neighbors not previously in a current frontier.

2. The graph processor of claim **1** wherein multiple of the first set of independent processing circuits identifies neighbors of different portions of the current frontier simultaneously and wherein multiple of the second independent processing circuits identifies portions of the new frontier simultaneously.

3. The graph processor of claim **2** wherein data accessed in the memory by the first set of independent processing circuits is disjoint in memory from data accessed by the memory by the second set of processing circuits for simultaneous access.

4. The graph processor of claim **2** wherein the first set of independent processing circuits and second set of independent processing circuits operate on frontiers of a single level of vertices at a time, where a level is determined by edge distance from a source vertex.

5. The graph processor of claim **4** wherein at a conclusion of calculation of the current frontier and new frontier for a level, the new frontier level determined by the second set of independent processing circuits becomes the current frontier used by the first set of independent processing circuits for a succeeding level.

6. The graph processor of claim **1** wherein the second set of independent processing circuits further writes result data acquired during the search to the memory.

7. The graph processor of claim **6** wherein the result data is distance from a source vertex at which the search started and a father vertex for each vertex.

8. The graph processor of claim **1** wherein the memory further holds a bit vector describing the current frontier and a bit vector describing a next frontier in the search and wherein the first set of independent processing circuits reads the current frontier bit vector to identify vertices in the frontier and the second set of independent processing circuits writes to the new frontier bit vector based on the identified neighbors that have not been searched.

9. The graph processor of claim **8** wherein the second set of independent processing circuits further writes result data associated with each vertex in the next frontier and determines which of the identified neighbors have not been searched using the result data for those vertices.

10. The graph processor of claim **8** further including a local memory smaller in size than the memory and holding a compressed bitmap smaller than the current frontier bitmap having bits representing a compressed form of the frontier bitmap and wherein the first set of independent processing circuits identifies neighbors to a current frontier of the search by first accessing the compressed bitmap and then accessing the frontier bitmap only if the compressed bitmap indicates a set bit in the current frontier bitmap.

11. The graph processor of claim **10** wherein the bits of the compressed bitmap represent the logical OR of bits of the current frontier bitmap.

12. The graph processor of claim **10** wherein the second set of independent processing circuits operates to atomically update the new frontier bit vector and the compressed bit vector.

13. The graph processor of claim **1** wherein the memory provides at least sixteen independent full-duplex channels for reading and writing.

14. The graph processor of claim **13** wherein the channels communicate serial packets tagged with respect to the independent processing circuit making and access request.

15. The graph processor of claim **13** wherein the number of independent processing circuits is no less than the number of independent memory channels.

6. The graph processor of claim **13** wherein the memory provides for atomic read/write operations.

17. The graph processor of claim **13** wherein the memory is a three-dimensional memory employing through-silicone-vias.

18. The graph processor of claim **13** wherein the memory provides an access page size of less than 128 bits.

* * * * *