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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
23 January 2014 (23.01.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/014906 A2

(51) International Patent Classification:
G06F 19/00 (2011.01)

(21) International Application Number:
PCT/US2013/050673

(22) International Filing Date:
16 July 2013 (16.07.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/672,028 16 July 2012 (16.07.2012) US

(63) Related by continuation (CON) or continuation-in-part (CIP) to earlier application:
US 61/672,028 (CON)
Filed on 16 July 2012 (16.07.2012)

(71) Applicant: **PNEURON CORP.** [US/US]; 99 North Eastern Blvd., Suite 100, Nashua, New Hampshire 03062 (US).

(72) Inventors: **MOSS, Simon, Byford**; 210 Bible Street, Greenwich, Connecticut 06807 (US). **ELKINS, Elizabeth, Winters**; 1255 North Shore Drive, Roswell, Georgia 30076 (US). **BACHELOR, Douglas, Wiley**; 26 Brown Lane, Groton, Massachusetts 01460-1485 (US). **CURBELO, Raul, Hugh**; 174 Cedar Street, Sturbridge, Massachusetts 01566 (US). **FOUNTAIN, Thomas, C.**; 86 Noe Avenue, Madison, NJ 07940 (US).

(74) Agents: **LYMAN, Beverly A., Ph. D.** et al.; Thompson Hine LLP, 10050 Innovation Drive, Suite 400, Dayton, OH 45342-4934 (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, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, 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).

Published:

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

(54) Title: A METHOD AND PROCESS FOR ENABLING DISTRIBUTING CACHE DATA SOURCES FOR QUERY PROCESSING AND DISTRIBUTED DISK CACHING OF LARGE DATA AND ANALYSIS REQUESTS

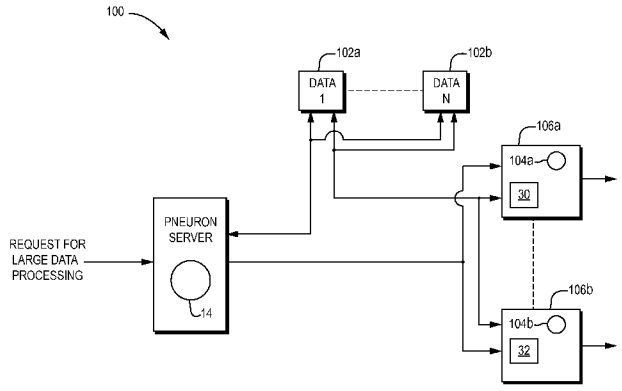

FIG. 4

(57) Abstract: A method and system for large data and distributed disk cache processing in a Pneuron platform (100). The system and method include three specific interoperable but distributed functions: the adapter/cache Pneuron (14) and distributed disk files (34), a dynamic memory mapping tree (50), and distributed disk file cleanup (28). The system allows for large data processing considerations and the ability to access and acquire information from large data files (102) and rapidly distribute and provide the information to subsequent Pneurons (104) for processing. The system also provides the ability to store large result sets, the ability to deal with sequential as well as asynchronous parallel processing, the ability to address large unstructured data; web logs, email, web pages, etc., as well as the ability to handle failures to large block processing.

WO 2014/014906 A2

A METHOD AND PROCESS FOR ENABLING DISTRIBUTING CACHE DATA SOURCES FOR QUERY PROCESSING AND DISTRIBUTED DISK CACHING OF LARGE DATA AND ANALYSIS REQUESTS

TECHNICAL FIELD

[0001] The present invention relates to enabling distributing cache data sources for processing large data and analysis requests and more particularly, relates to providing a distributed caching model to enable the management of distributed cache files on multiple servers or virtual machines and facilitating multiple distributed processing operations simultaneously.

BACKGROUND INFORMATION

[0002] Accessing geographically dispersed multiple systems and large datasets and being able to operate on this information to perform multiple simultaneous operations is very difficult. Combining and federating distributed operation results together compounds the problems. Most companies utilize an aggregated data warehouse with multiple feeder data sources and extraction, transformation, and loading (ETL) routines to organize distributed data together. The data preparation cost and time are signification.

[0003] Therefore, what is needed is a distributed cache evaluation and processing model that operates across multiple servers simultaneously. The system should function such that multiple analytic and business operations occur, while the system should also enable sampling and evaluation with collection and recording of results. Furthermore, the invention should provide for distributed cache creation and orchestration of coordinated distributed data access and

generation of iteration results from other distributed applications. All distributed cache files operations should be coordinated together into unified processing models.

SUMMARY OF THE INVENTION

[0004] The system and method of the present invention implements an Adapter Pneuron that interacts within its distributed processing infrastructure for large data processing. The Adapter Pneuron enables the real-time acquisition of data from different types of application data sources, including service application programming interface (API), database, and files. Data is acquired and transformed into self-describing ASCII disk cache files with an associated definition of the structure. The disk cache files are distributed across one to many servers or virtual machines (VMs). The distributed disk cache files are accessed by participating Pneuron applications to perform operations selectively on the distributed disk data. Multiple operations are performed simultaneously by the different Pneurons with results evaluated and subsequent iteration operations applied. Evaluated results are concatenated and federated together across the different disk cache files simultaneously.

[0005] Disk cache files are removed automatically using a high-low disk evaluation model to remove disk cache files based on server disk utilization and automatic evaluation aging for disk cache files. The present invention enables the ability to quickly access target systems and data sources and generate distributed disk cache files, to simultaneously perform real-time operations by other Pneuron programs and to federate the results together. These activities occur without requiring preparation of the data.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] These and other features and advantages of the present invention will be better understood by reading the following detailed description, taken together with the drawings wherein:

[0007] FIG. 1 is a comparison of the prior art process execution with the distributed cache model according to one embodiment of the present invention;

[0008] FIG. 2 is an overview of the dynamic memory mapping tree according to one embodiment of the present invention;

[0009] FIG. 3 is an overview of distributed disk cache removal model scenarios according to one embodiment of the present invention; and

[0010] FIG.4 is a block diagram of a system on which may be implemented the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0011] The present invention features a system and method for large data processing and requesting reconstruction. The system 100, FIG. 4 and method includes the capacity for large data processing considerations (targeting record, queries and responses of 1 million and higher results). The invention provides for the ability to access and acquire information from large data files 102 (greater than 1 million records) and rapidly provide the information to subsequent Pneurons for processing combined with the ability to extract and render large queries from databases 102 without impacting system of records processing and rapidly provide the information to subsequent Pneurons for processing. The system also has the ability for multi-threaded processing by multiple distributed pneurons 104 of large input/record sets

files, enabling of storage and access to large historical results and the ability to handle large inputs. The invention provides the ability to store or persist large result sets. For example, a million plus raw data evaluation may generate a very large array of intelligence results that need to be persisted for future use, which might occur with time-series data with multiple month-years and multiple intelligence results for each intelligence record. Further, the invention is able to deal with sequential as well as asynchronous parallel processing, is able to address large unstructured data; web logs, email, web pages, etc. and is able to handle failures to large block processing.

[0012] The design considerations of the present invention are focused on maximizing distributed processing workload (volumes, results and requests) without running out of resources; e.g. hardware resources, including memory and CPU. The solution consists of essentially three specific interoperable but distributed functions. First, the Adaptor/Cache Pneuron 14 and distributed disk cache files 30, 32. Second, the dynamic mapping tree 50, Fig. 3. Third, the distributed disk cache file cleanup Fig. 4. Each function will be described in greater detail below.

[0013] The Adaptor/Cache Pneuron 14 (and/or distributed adaptor/cache pneurons 104) and distributed disk cache files 34 address the problem of extremely large record set processing which presents different technology challenges. Some of the current problems in the prior art include: loading all information into memory will exceed hardware server resources; and breaking up large requests presents complexities in consolidating and synchronizing the information results together and multiple operations may be

required at different times by different programs across one or more large record sets.

[0014] The present invention solves these problems by extracting large record sets from target systems 102 and data sources and converting them into distributed disk cache files 34. The disk-based intermediate cache files and processing is coordinated by and across multiple Pneurons 104 to perform multiple simultaneous operations on the information (distributed disk cache files 34). A comparison of the prior art system of process execution (FIG. 1A) and the distributed cache model of the present invention (FIG. 1B) is shown in Figure 1.

[0015] The cache file based system 10 of the present invention will store the large requests within self-describing ASCII files and make these files (the data within them) available to any Pneuron that needs to access them. Large data requests 12, are received and processed by the Adapter Pneuron 14. The Adapter Pneuron 14 transforms the large data requests into ASCII file content (extended CSV format - including the attribute type definition), and saves the ASCII file content on the local host hard drive. Once a request is received, the Adapter Pneuron 14 will send to all its associated Pneuron connections 104 a special message that will announce that new work is available and the data can be accessed from the referred files from the target disk cache location 30, 32 on the file system. This process will perform in the same manner even if the request is composed from multiple batches, thereby allowing the request to be reconstructed. All of the Pneurons will interact with this model approach. The Adapter Pneuron 14 maintains context of each distributed cache file and provides system context to each participating Pneuron. Context includes the definition

of the cached file format and information elements and location of the file. Participating Pneurons are able to parse the cached/adaptor Pneuron information and perform different operations.

[0016] Once the data has been cached, the Adapter Pneuron 14 will send to subsequently connected Pneurons 104 a special message 15 that will announce to all configured and associated Pneurons that new work is available and the Pneurons can execute their operations on the disk cache files. The system includes a utility that enables the calling Pneurons 104 to transform to and from XmlMessage to the target extended CSV extended file format.

[0017] As a result, the invention greatly simplifies the access and operations on the distributed disk cache data and provides a common abstraction layer and interface for the Pneurons to access and perform operations on the data. The Pneurons only need to read the referred file content and transform the information into usable XmlMessage Type data. In addition, the Pneurons can filter and extract only the necessary attributes as vectors or other objects and optimize the memory management resources.

[0018] This invention therefore provides many critical transformational benefits. The data is accessed and managed at targeted server 106 locations on the respective filing system 30, 32 such that requests do not need to be reconstructed, which saves processing time and reduces complexity. The system ability to process very large amounts of data is significant and unconstrained. Within the actual memory processing, the information is streamlined. Only reference and common messages and pointers are included. The distributed cache file model enables a planning mechanism to be implemented to optimize the resources and synchronize

distributed cache file access and processing. The messages do not require any complex logical operations that will require the file structure to change. The system will be fully capable of handling the CRUD operations (create - add new entry/record; read - record; update - record; and delete - record). This solution will work for all cases where the entity (large request - as a whole) will retain its integrity/structure.

[0019] The dynamic mapping tree model shown for example in FIG. 2 is implemented to support the Adaptor Pneuron. The memory mapping enables a large data processing request transaction to retain its processing integrity from initiation through completion of an execution. By retaining processing integrity, the representation and all the data characteristics will be retained and accessible during the request life cycle. Data representation defines the meta-data characteristics of the information, including the way that the data is stored on the file system, the number of files, file types, data definition (attribute definition), request references etc.

[0020] In order to manage the distributed disk caching model, the invention enables the following operations to be performed on the disk cache files: **C**reate - add new record within the large request; **R**ead - access one or more records from the large request; **U**pdate - update/modify the data for one or more records; and **D**elete - delete one or more records. Given the synchronization and management complexities, the invention restricts the following functions: batching, duplicate batches and conditional batches.

[0021] To manage the distribution complexity of multiple disk cache files, the invention maintains and adjusts the system context dynamically. This model enables automatic

changes to the data representation and structure. A programmatic change history tracking is maintained, which keeps track of changes applied to the disk cache file(s). This feature enables automatic reconstruction of the disk cache file at any given time to support a Pneuron initiated operation and request. The present invention has implemented a programmatic process to decompose large data sets into request into smaller batches. The batches are organized into parallel execution requests and configured as part of the Pneuron Networks definition.

[0022] A dynamic memory tree map, Figure 2, is implemented to manage the distributed cache process across multiple Pneurons. The dynamic tree maintains and provides system context for the entire distributed processing model and plan. The entire processing life cycle is maintained. Each node/leaf within the dynamic tree will contain a file reference or a position/index reference and then point the Pneuron request message to the corresponding memory area. The dynamic memory tree map establishes a breadcrumb trail. Using this approach, the system is able to reconstruct the request with the new values by traversing the memory tree. The system merges and reconstructs the disk cache results based on the specific request. The same logic and approach is also applied for the Large Request Reconstruction, which enables a generic distributed disk cache operation model to be applied at the Pneuron Base Level.

[0023] The system will apply different solutions based on the context and type of operation. Dead (empty) messages are still sent out through the network. When a batch gets split in two or more sub-batches they are flagged. By doing this the system will be able to track the messages. The final Pneuron should have a max dead time interval, which will

represent the time that it will wait for more batches. This time is checked/validated with the last batch arrival time. Each time a batch gets split the characteristic flag is appended with additional information meant to inform about the split. Example: 1/1-3/15-1/3-6/7-4/4. SPLIT is defined as [Position/Number Of Message/Batch] / [Total Number Of Messages]. Each time a batch gets split request, the split information will be appended to the current flag, which will be done for each split/sub batch. By the time the message reaches the Final Pneuron, the Pneuron will be able to establish the context based on the amount of information that it receives, and the Pneuron will be ready to create an execution tree, such as the one detailed in Figure 2. This approach is based on the fact that when the Final Pneuron receives a batch request, it will be able to trace it and complete (or start if it is the first batch from a large request) based on the defined execution tree. Any sub-batch that is received is able to communicate with the Pneuron of all the tree node parents and also the number of "leafs" per split. With this approach the Final Pneuron will be able to map out what it should receive, also the information that it receives can be ordered.

[0024] There are scenarios where the requesting Pneuron is unable to interact with the distributed cache disk. Examples could include: (1) The target data source or system is not available for access by the Adapter Pneuron and the disk file cache cannot be created; and (2) The file system where the disk cache file is stored is not available. An Idle or Dead Time interval model can be implemented to manage this scenario, such that the Idle or Dead Time interval establishes a periodic mechanism to compose the message and send it further (or execute the request). The Idle or Dead

Time interval evaluates each past request and the elapsed time when the last batch was received and the execution trigger.

[0025] Finally, the distributed disk cache file clean up portion of the process 28, Figure 3, provides users with the capability of caching data, within the entire system, on all the hosts 106 that are running the platform (distributed pneuron 104). The cache is a file system 34 based mechanism that transforms and stores them indefinitely making them available to one or more worker process pneurons. Since the invention is dealing with a highly distributed system that provides value by providing the users with parallel computing capabilities, all the resources that are used within this computing process must be available at each host level (that takes part of the parallel execution). In doing so, each host will own a copy for each cache data that it will process. This creates a big problem because the hardware resources, hard drive space in this case is not unlimited, and since each host must have a local copy of the cached job the system does not deal with replication (duplicate resources - at different host levels).

[0026] Therefore, the present invention has implemented a High-Low distributed disk cache removal model. The invention configures properties for each host 106 (either a physical server or virtual server machine). The host Max Available Space property establishes the amount of bytes (megabytes or even gigabytes) that can be used by the caching system 34 on that specific server 106. Once this max threshold is reached, the system will delete existing cache files based on the size and age of the distributed cache file. This model will eliminate past files and enable new disk files to be established and used. The cache file system will be bounded

with these rules; in this case the only rule/limitation that we need is to have a maximum level of space that it can be used in order to store the working cache files. This maximum level of space that can be used will be stored within the Softwarerx.Properties file 36 from CFG directory, because this is a centralized storage point for all the properties and attributes that must or can't be stored within the database.

[0027] The following examples are intended to provide details on how the distributed disk file clean up functions in the present system. In a first example, a save cache data request 38 is requested/received and max space has not been reached on the host server 30/32. In this scenario, a Pneuron issues a request 38 to save data into the cache data file system 34. The request reaches the SAN (Storage Area Network or Cache System/process) 40. The system checks the Max Space configured value 36. The system 28 compares the Max Space with the actual available space on the local hard drive, which is the hard drive where the host system 106 is running, or more exactly where the "cache" directory file system 34 is found. In this first example there is sufficient space to save the information; therefore the system 28 will save the information 42 with the provided data (reference name/file name) in the file system 34.

[0028] In a second example, a save cache data request is requested and max space has been reached. In this scenario, a Pneuron issues a request to save data into the cache data system. The request reaches the SAN (Storage Area Network or Cache System). The system checks the Max Space configured value. The system compares the Max Space with the actual available space on the local hard drive, which is the hard drive where the system is running, or more exactly where the

"cache" directory is found. The system determines 44 there is NO sufficient space to save the information. The system orders the existing cache data in descending order based upon the creation date. Then a loop occurs, which deletes the oldest file 46 and then re-checks to see if there is sufficient space. The loop ends once sufficient space is cleared or if there is nothing else to delete. If the system has sufficient space to save, then the information is saved 42 with the provided data (reference name/file name).

[0029] In a third example, a save cache data request is requested and max space has been reached, however the system is unable to make sufficient space. In this scenario, a Pneuron issues a request to save data into the cache data system. The request reaches the SAN (Storage Area Network or Cache System). The system checks the Max Space configured value. The system compares the Max Space with the actual available space on the local hard drive, which is the hard drive where the system is running, or more exactly where the "cache" directory is found. The system finds there is NO sufficient space to save the information. The system orders the existing cache data descending based upon the creation date. A loop is created, such that the oldest file is deleted and then the system re-checks to see if there is sufficient space. In this example, the system deletes all old files 46 and checks again for sufficient space and determines that there is not sufficient space and there is nothing else to delete, thereby ending the loop. In this example, the system does not have sufficient space to save and the system will register a failure.

[0030] In a fourth example, a system is able to get cache data when a local copy is available. In this scenario, the cache system receives a request 48 to get a specific data.

This request can be issued by any Pneuron Instance that is supposed to use the cached data and needs to get a reference to the local file copy in order to read and parse/analyze or otherwise utilize the necessary information. The system receives a request to get cache data 48. The system process cache 50 checks to see if the cached data is found within the local file system 34. The cache data is found to exist 52 within the local file system. Return reference to cache data 54. The caller will then be able to use the data.

[0031] In a fifth example, a system is unable to get cache data because a local copy is not available. In this scenario, the cache system 30, 32 receives a request to get specific data 48. This request can be issued by any Pneuron Instance that is supposed to use the cached data and needs to get a reference to the local file copy in order to read and parse/analyze or otherwise utilize the necessary information. The system receives a request to get cache data 48. The system cache process 50 checks to see if the cached data is found within the local file system 34a. The system determines that the cache data DOES NOT EXIST within the local file system. The Current Cache System asks the other registered host 32 by calling their associated cache system process 50a which check for existence of the data. A loop is created, such that the Foreign Cache file system 34b of server 32 is checked for data 56, then the data is found, and then the data is copied locally 58. The loop ends when there are no more hosts/cache systems to search or once the cache data is found. Return reference to cache data 58. The caller host 30 will then be able to use the cached data.

[0032] In a sixth example, a system is unable to get cache data because a local copy is not available anywhere. The cache system receives a request to get a specific data. This

request can be issued by any Pneuron Instance that is supposed to use the cached data and needs to get a reference to the local file copy in order to read and parse the necessary information. The system receives a get cache data request. The system checks to see if the cached data is found within the local file system. The system determines that cache data DOES NOT EXIST within the local file system. The Current Cache System asks the other registered host by calling their associated cache systems 32 and checking for the data existence. A loop is created, wherein the system checks the Check Foreign Cache System for data and determines that the data is not found. The loop ends once there are no more hosts/cache systems to check and no cache data has been found. The system determines that the data was not found. A failure has occurred.

[0033] In summary, the present invention enables the real-time generation, management, and synchronization of distributed disk caches within a highly distributed processing environment. The process deconstructs and organizes large data sets acquired from disparate systems and data sources across an unlimited number of physical servers and virtual machines. An abstraction layer is applied across all distributed disk cache files. Multiple distributed Pneurons perform simultaneous operations across one or more disk cache files. Processing is synchronized automatically. The system maintains an in-memory mapping tree to maintain distributed interactions and provides the ability to dynamically construct and deconstruct the distributed cache files into any form. The distributed cache model enables synchronized federation of selected information from multiple distributed cache files automatically and as part of the Pneuron processing. The invention allows Pneuron to use

existing client disk capacity and obtain and utilize targeted large data cache files on demand and without preparing aggregated data stores. As a result, businesses benefit by foregoing large data preparation activities.

[0034] Modifications and substitutions by one of ordinary skill in the art are considered to be within the scope of the present invention, which is not to be limited except by the allowed claims and their legal equivalents.

CLAIMS

The invention claimed is:

1. A method for processing large data and analysis requests, said method comprising the following acts:

acquiring data in real-time from one or more application data sources by a data processing pneuron, said data processing pneuron configured for operating under control of a computer program product, said computer program product comprising a computer program;

transforming said data into self-describing ASCII disk cache files by said data processing pneuron;

distributing said disk cache files across one or more servers or virtual machines by an adaptor cache pneuron;

allowing access to said distributed disk cache files by participating applications, wherein said applications are configured to perform operations on said distributed disk data; and

removing said disk cache files automatically using a high-low disk evaluation model to remove said distributed disk cache files based on server disk utilization and automatic evaluation aging for said distributed disk cache files.

2. A system for processing large data and analysis requests, said system comprising:

a pneuron server configured for operating under control of a computer program product, said computer program product comprising a computer program, for deploying a data processing pneuron for acquiring data in real-time from one or more application data sources, said data processing pneuron for transforming said data into one or more self-describing ASCII disk cache files;

said pneuron server configured for deploying one or more adaptor cache pneuron, said adaptor cache pneuron operating under control of a computer program product, said computer program product comprising a computer program, for distributing said one or more disk cache files across one or more servers or virtual machines;

said one or more servers configured for allowing access to said distributed disk cache files by one or more distributed worker pneurons, wherein said distributed worker pneurons are configured to perform operations on said distributed disk data; and

said one or more worker pneurons configured removing said disk cache files automatically using a high-low disk evaluation model to remove said distributed disk cache files based on server disk utilization and automatic evaluation aging for said distributed disk cache files.

3. The system of claim 2 wherein said one or more worker pneurons configured removing said disk cache files are configured for removing disk cache files prior to storing disk cache files received from said adaptor pneuron.

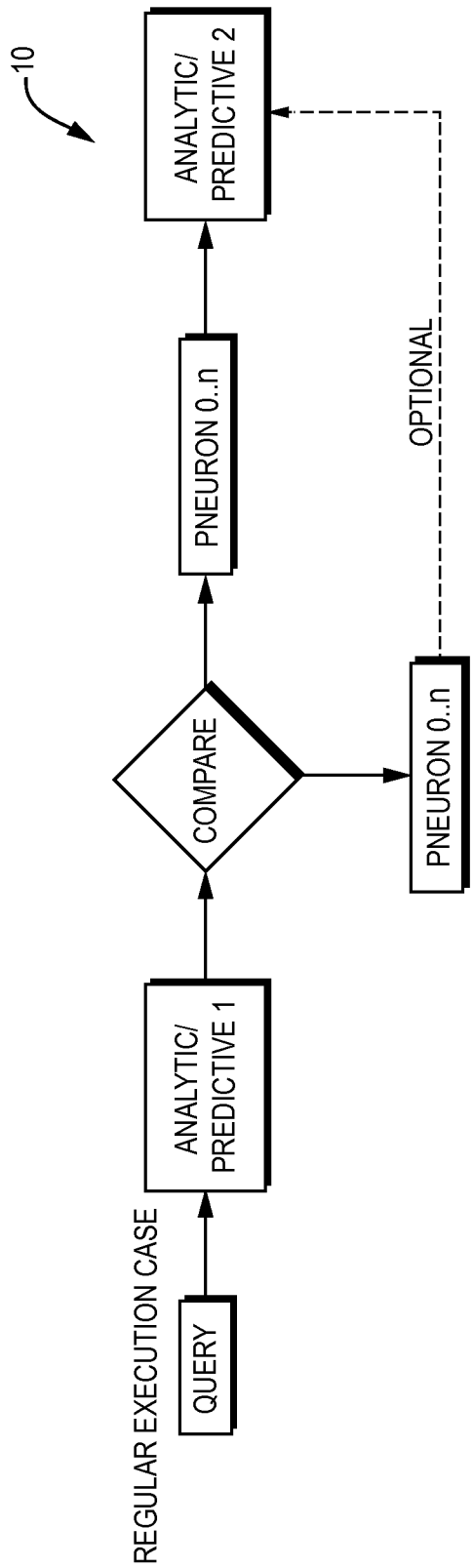
10
REGULAR EXECUTION CASE
QUERY
ANALYTIC/ PREDICTIVE 1
COMPARE
PNEURON 0..n
OPTIONAL
PNEURON 0..n
ANALYTIC/ PREDICTIVE 2

FIG. 1A

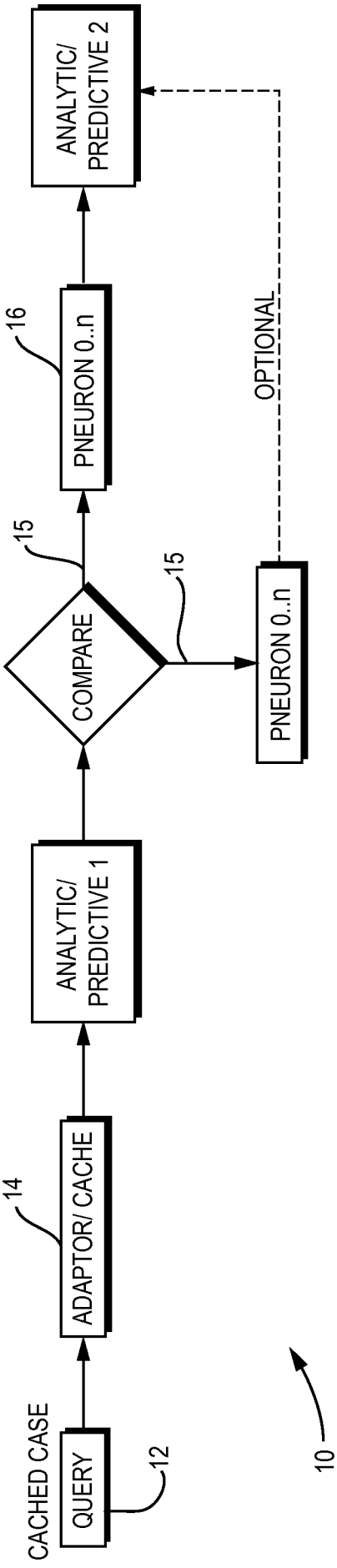
CACHED CASE
QUERY
12
ADAPTOR/ CACHE
14
ANALYTIC/ PREDICTIVE 1
15
COMPARE
15
PNEURON 0..n
OPTIONAL
16
PNEURON 0..n
ANALYTIC/ PREDICTIVE 2
10

FIG. 1B

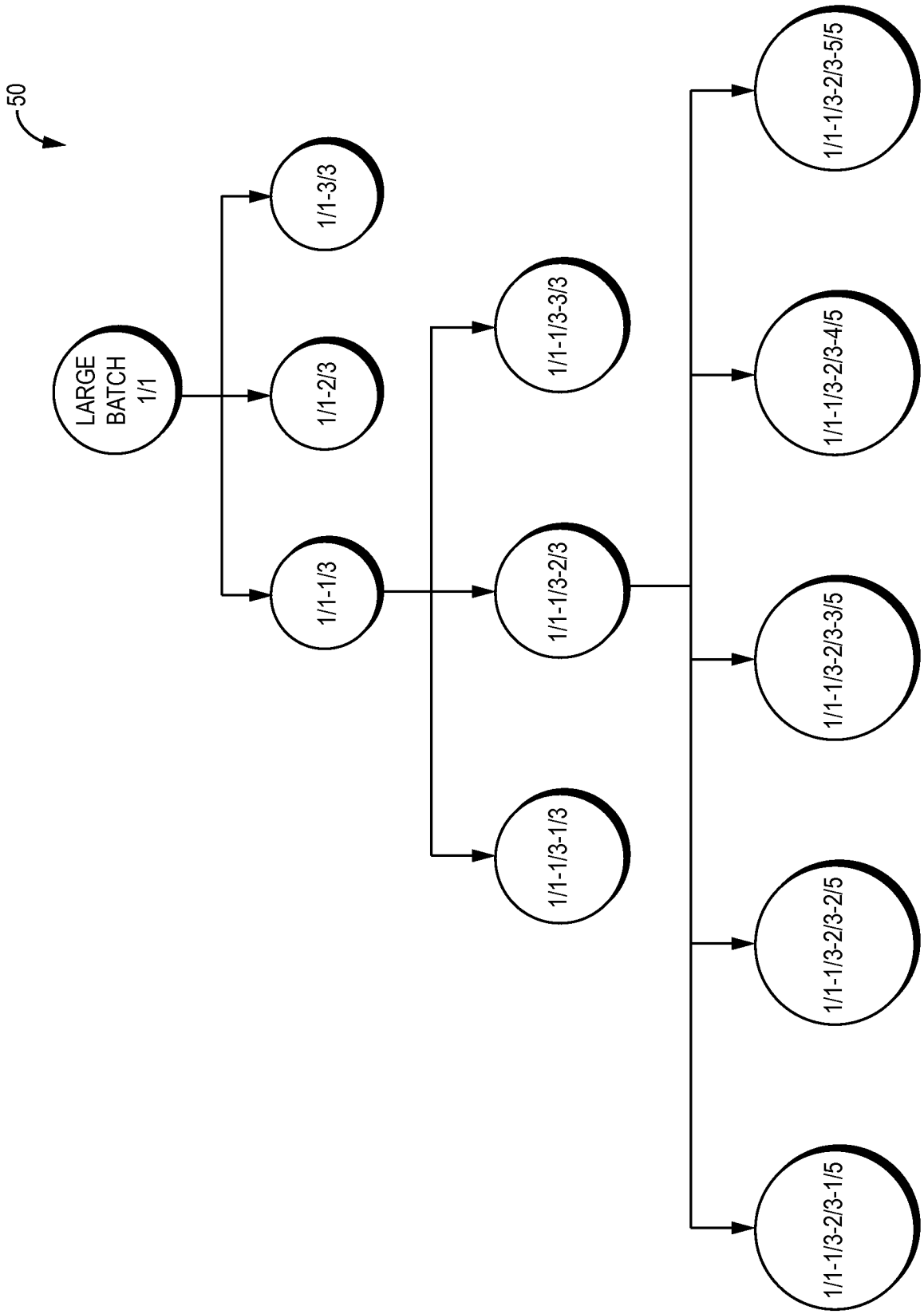
50
LARGE BATCH 1/1
1/1-3/3
1/1-2/3
1/1-1/3
1/1-1/3-3/3
1/1-1/3-2/3
1/1-1/3-1/3
1/1-1/3-2/3-5/5
1/1-1/3-2/3-4/5
1/1-1/3-2/3-3/5
1/1-1/3-2/3-2/5
1/1-1/3-2/3-1/5

FIG. 2

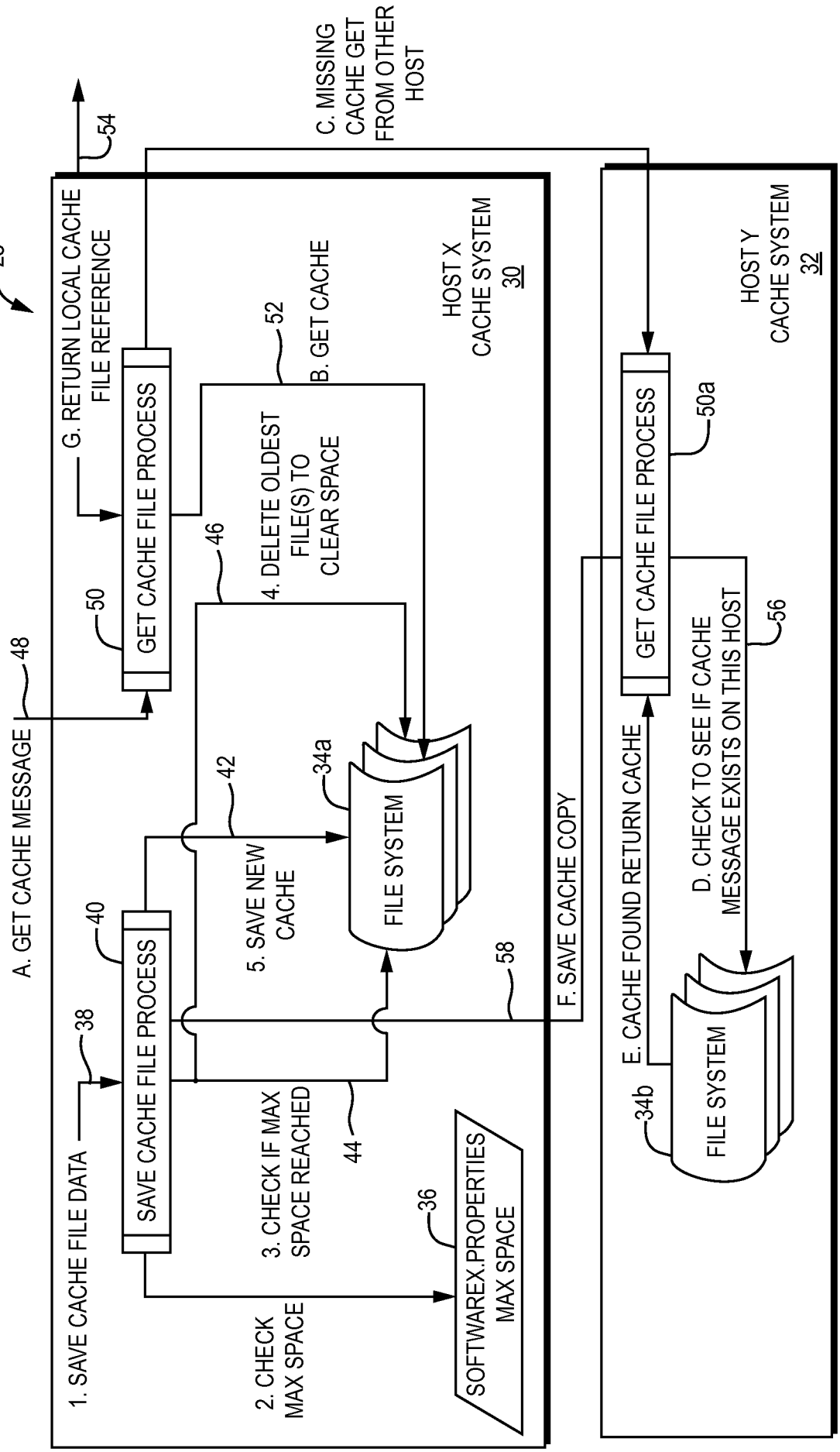
28
A. GET CACHE MESSAGE
48
GET CACHE FILE PROCESS
50
G. RETURN LOCAL CACHE FILE REFERENCE
54
C. MISSING CACHE GET FROM OTHER HOST
HOST X CACHE SYSTEM 30
B. GET CACHE
52
4. DELETE OLDEST FILE(S) TO CLEAR SPACE
46
5. SAVE NEW CACHE
42
34a
FILE SYSTEM
SAVE CACHE FILE PROCESS
40
1. SAVE CACHE FILE DATA
38
3. CHECK IF MAX SPACE REACHED
44
2. CHECK MAX SPACE
SOFTWAREX.PROPERTIES MAX SPACE
36
58
F. SAVE CACHE COPY
HOST Y CACHE SYSTEM 32
GET CACHE FILE PROCESS
50a
E. CACHE FOUND RETURN CACHE
D. CHECK TO SEE IF CACHE MESSAGE EXISTS ON THIS HOST
56
34b
FILE SYSTEM

FIG. 3

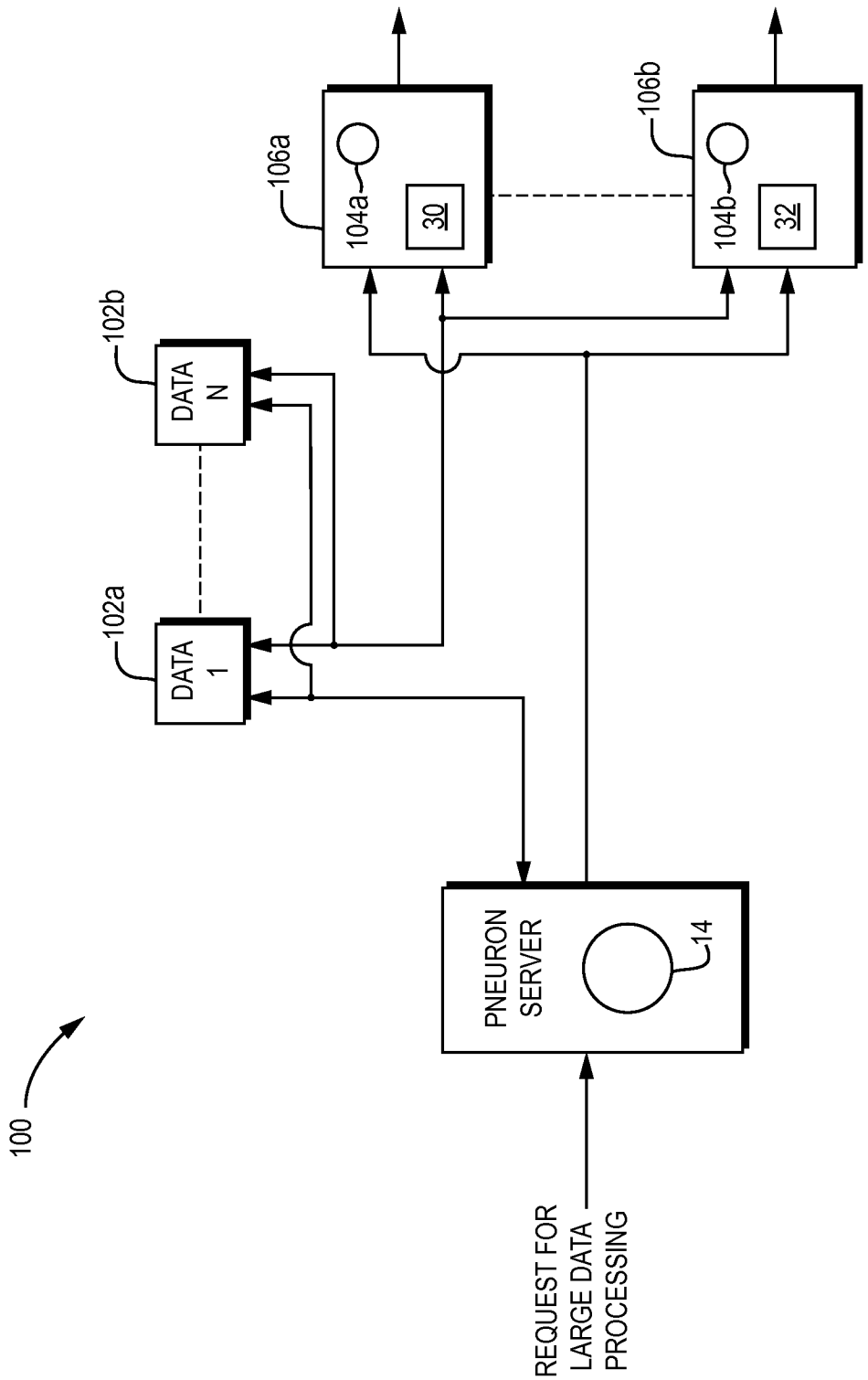
100
PNEURON SERVER
14
REQUEST FOR LARGE DATA PROCESSING
DATA 1
102a
DATA N
102b
104a
30
106a
104b
32
106b

FIG. 4