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

(11) EP 3 547 102 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
02.10.2019 Bulletin 2019/40

(21) Application number: **19159540.4**

(22) Date of filing: **26.02.2019**

(51) Int Cl.:
G06F 3/06 $^{(2006.01)}$ ***G06F 16/13*** $^{(2019.01)}$
G06F 12/0864 $^{(2016.01)}$

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

(30) Priority: **29.03.2018 US 201815940961**

(71) Applicant: **INTEL Corporation**
Santa Clara, CA 95054 (US)

(72) Inventors:
- **KUMAR, Mohan J.**
  Aloha, OR Oregon 97007 (US)
- **CHAGAM REDDY, Anjaneya R.**
  Chandler, AZ Arizona 85226 (US)

(74) Representative: **Rummler, Felix**
Maucher Jenkins
26 Caxton Street
London SW1H 0RJ (GB)

(54) OBJECT STORAGE SYSTEM WITH MULTI-LEVEL HASHING FUNCTION FOR STORAGE ADDRESS DETERMINATION

(57) A method performed by a first hardware element in a hierarchical arrangement of hardware elements in an object storage system is described. The method includes performing a hash on a name of an object of the object storage system. The name is part of a request that is associated with the object. A result of the hash is to identify a second hardware element directly beneath the first hardware element in the hierarchical arrangement. The request is to be sent to the second hardware element to advance the request toward being serviced by the object storage system.

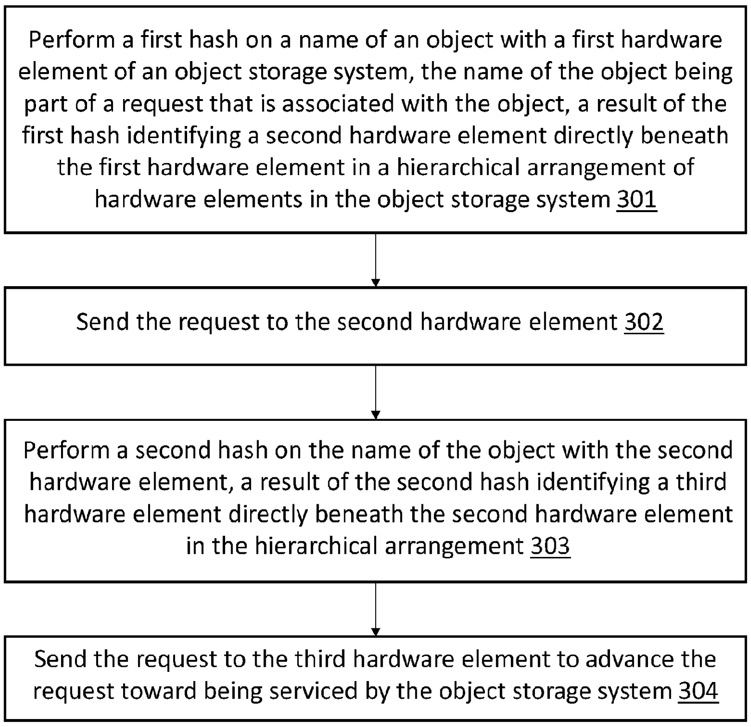

Perform a first hash on a name of an object with a first hardware element of an object storage system, the name of the object being part of a request that is associated with the object, a result of the first hash identifying a second hardware element directly beneath the first hardware element in a hierarchical arrangement of hardware elements in the object storage system 301
Send the request to the second hardware element 302
Perform a second hash on the name of the object with the second hardware element, a result of the second hash identifying a third hardware element directly beneath the second hardware element in the hierarchical arrangement 303
Send the request to the third hardware element to advance the request toward being serviced by the object storage system 304

Fig. 3

Description

Field of Invention

[0001] The field of invention pertains generally to the computing sciences, and, more specifically, to an object storage system with multi-level hashing function for storage address determination.

Background

[0002] With the emergence of cloud computing and high performance data centers as an infrastructure environment for modern day computing, innovations are being made in large scale data storage. One particular emerging storage approach, referred to as object storage, is characterized by a highly versatile input namespace, ease of operation with distributed storage nodes and corresponding large scale storage capacity. When attempting to physically build the hardware and/or software needed to implement an actual, working object storage system, however, issues of practical implementation may be observed resulting in an opportunity for practical innovation improvements.

Figures

[0003] A better understanding of the present invention can be obtained from the following detailed description in conjunction with the following drawings, in which:

Fig. 1 shows a prior art object storage system;
Fig. 2 shows an improved object storages system;
Fig. 3 shows a method performed by the improved object storage system;
Fig. 4 shows a computing system.

Detailed Description

[0004] An issue with object storage systems is the movement of data objects as a consequence of storage nodes being added or deleted from the system. Here, as observed in Fig. 1, an object storage system 100 is typically implemented with a number of storage nodes 101 that are communicatively coupled to a network 102. Each storage node contains one or mass storage devices having associated storage locations and corresponding storage location addresses.

[0005] When a client submits an object for storage in the system 100, the system performs a hashing operation upon the name given to the object by the client (e.g., a directory file path for the object, a unique identifier assigned to the object, etc.). The output of the hashing operation essentially generates address of the storage location in the system where the object is to be stored. Each of the storage nodes in the system 101 is assigned a different range of object storage location addresses. If a new storage node is added to the system or deleted from the system, the per storage node object address location assignments change which requires the movement of some objects from their present storage node to a new storage node.

[0006] Fig. 2 shows an improved object storage system design 200. According to the object storage system design of Fig. 2, the architecture of the system takes the form of a nodal tree in which each higher level node of the tree subsumes the lower level nodes beneath it. The lowest level 204 of the tree corresponds to leaf nodes having the actual storage media devices (e.g., flash drives, hard disk drives, etc.). Additionally, each nodal level is preceded with a dedicated hashing function for that nodal level that identifies a particular node at the nodal level to which a particular object, based on its name, is to be mapped. For simplicity Fig. 2 shows an object storage system 200 with only three nodal levels 201, 202, 203 (and therefore three hashing functions 211, 212, 213 that precede them). However, the reader will understand that additional nodal levels and corresponding hashing functions can be added to the three-level system of Fig. 2.

[0007] Referring to Fig. 2, when a new object is to be stored in the system, the first hashing function 211 determines to which of the highest level nodes 201 the object is to be mapped and the object is sent over a network 204 to that node. Each of the highest level nodes 201 include an instance of a second hashing function 212. The particular highest level node that the object mapped to computes a second hash on the object name with the second hashing function 212. The result identifies a node at the next lower level 202 that the object maps to and the object is sent over a network to that node. The next lower level node also includes an instance of a third hashing function 213. The particular next lower level node that the object mapped to computes a third hash on the object name. The result identifies a node at the lowest, leaf node level 203 that the object maps to. The object is sent to that leaf node and is stored therein.

[0008] Additionally, each of the hashing algorithms 211, 212, 213 are weighted to have some bias, or will otherwise assign objects to immediately lower nodes in a purposely imbalanced way, if the structure of the object storage system beneath the hashing function is likewise biased or imbalanced. For example, if a first immediately lower node has more storage capacity beneath it and/or more I/O bandwidth than a second immediately lower node, the hashing function that feeds the first and second nodes will be structured to map more objects to the first node than the second node.

[0009] Here, each hashing function 211, 212, 213 is presented with some description of the storage system structure directly beneath it. The description essentially describes, e.g., the total storage capacity, bandwidth capacity, media type, processor speed, etc. at and/or beneath each node that it maps objects to. The description may be automatically calculated from child node descriptions. In further embodiments an administrator or administration function can override automatic description settings to meet various service level agreements.

[0010] In the case of higher level hashing algorithms that map objects to higher level nodes, the storage capacity of a higher level node corresponds to the combined storage capacity of the leaf nodes beneath it. Thus, for instance, the storage capacity of Rack_1 corresponds to the combined storage capacity of storage server nodes SS_1_1 through SS_1_N. The description of the performance capability of one or more of the nodes directly beneath a hashing level can also be referred to, in various embodiments, as a "weight factor". Hence, each hashing level 211, 212, 213 receives one or more weight values describing the capability of the storage system nodes directly beneath it so that it can weigh or bias object storage location assignments in response to any imbalances in the capabilities of the nodes directly beneath it if such imbalances exist.

[0011] In further embodiments, seed values may be assigned to each node in the hierarchical tree of the object storage system. Seed values are randomized numbers that are used for the hashing calculations. In various embodiments, the aforementioned one or more weight factors are presented as input value(s) to a random number generator at a particular node. The random number generator then generates a seed value for the node's hashing function. In this manner, each node in the tree generates a seed value and then uses the seed value to perform hashes with the object name of each request received by the node to determine a next lower node that the request is to be directed to.

[0012] The seed value generated at each node therefore includes some randomness owing to its being generated from a random number generator (if all lower nodes are equal, requests should be mapped randomly to the lower nodes). The weights therefore provide some bias to the randomness of the seed values that causes the hashing function to favor one node (e.g., having more storage capacity) over another node (e.g., having less storage capacity) when determining which lower node a request it to be mapped to. In alternate embodiments, weight factors may be sent to a centralized location (e.g., that maintains an understanding of the object storage system's hierarchical tree) having a random generator that calculates seed values for multiple nodes of the system. The centralized location may then send the appropriate seed values to the different nodes for incorporation by their respective hashing functions.

[0013] In an embodiment, execution of any particular one of the hashing algorithms with its corresponding object name, input weight factor(s) and seed value generates a ranking of all the nodes directly beneath the hashing level. The hashing algorithm then picks the highest ranked node as the node to which the object should be mapped (i.e., the object's assigned storage location falls within the range of location addresses assigned to the selected node).

[0014] In various embodiments, each nodal level corresponds to a different element of hardware in the overall object storage solution. The highest nodal level 201 in the exemplary system of Fig. 2 corresponds to a rack of storage servers. Here, as is known in the art, a rack is typically implemented as one or more cabinets having shelf space or otherwise retrofitted to safely stack electronic equipment. In the case of an object storage system, the rack may stack storage servers, which, e.g., are servers that emphasize storage by, e.g., having most/all of the higher bandwitdh I/O interfaces of the server computer coupled to non volatile mass storage devices (e.g., a solid state drives (e.g., flash SDD) and/or hard disk drives).

[0015] In the exemplary system of Fig. 2, with the highest level nodes 201 corresponding to a different rack, then, each of the immediately lower level nodes 202 directly beneath a particular rack/node correspond to the different storage servers that are stacked in the rack. In turn, each leaf node beneath a particular storage server node in Fig. 2 corresponds to a particular non volatile mass storage device that is integrated with the particular server node (e.g., its coupled to one of the server's high bandwidth I/O interfaces). From this example, as the object storage system expands in scale and/or more granularized hardware boundaries are desired per nodal level, higher level nodes may be added above the highest level nodes.

[0016] For example, in the case of expansion, if the set of racks and corresponding equipment observed in Fig. 2 correspond to the set of equipment within a first data center, and then the system is expanded to include more data centers, higher level nodes can be added to those observed in Fig. 2 such that each added higher level node corresponds to a different data center.

[0017] In the case of more granularized hardware boundaries, for example, even if the equipment observed in Fig. 2 corresponds to the equipment within a same data center, different groups of racks may be coupled to different local area networks within the data center. As such, higher level nodes may be added where each added higher level node corresponds to the gateway to a specific local area network and the racks/nodes beneath each added higher level node correspond to the racks that are coupled to the local area network that is represented by the higher level node. Additional nodes may also be added below the rack level to introduce more finely granularized hardware boundaries within a rack. For example, a nodal level may be added between the storage server level and the mass storage device level to group those mass storage devices that are coupled to a same I/O interface of a particular storage server.

[0018] Regardless, note that the architecture may include the introduction of an object storage hashing function in various types of hardware resources for the purpose of mapping an object to a particular lower level hardware resource, where, the hardware resource is more of a component of the storage hardware infrastructure itself than a centralized object mapping function for the object storage system (whose purpose is to determine global object mappings that definitively define object storage locations from a single hashing operation).

[0019] That is, to some extent, the new approach of Fig. 2 can be viewed as taking a centralized object mapping function as described above with respect to Fig. 1 and not only distributing it, but distributing down through multiple, lower hierarchical levels of the storage hardware itself (e.g., a local area network gateway, a network gateway that is integrated into a rack, a storage server, an interface to a storage server I/O interface, etc.).

[0020] The distribution of the hashing function activity needed to fully resolve a final storage location for a particular object through different levels of the storage hardware helps insulate the overall storage system from object re-location after the addition or removal of a storage capacity. Here, a single global hashing function is apt to have names of objects resolve to different storage locations in a more disruptive way in response to the addition/removal of a storage node than an object storage system constructed according to the hashing structure of Fig. 2.

[0021] Qualitatively describing the mathematical behavior, if a single hashing function is solely responsible for determining the storage locations of all objects in the system, a single change to the storage capacity of the system (even a small one) may cause large numbers of object names to suddenly map to physically distant new locations (e.g., to a new local network, to a different rack, etc.). To suppress this particular effect, object storage systems have traditionally been implemented with consistent hashing algorithms which mathematically have reduced numbers of mapping changes in response to storage capacity change than other hashing functions. Conceivably, however, the resulting relocations may still require significant physical transportation of objects across the system.

[0022] However, the object storage system of Fig. 2, which implements the overall hashing activity as a sequence of hashing processes that "thread" from higher hardware levels to lower hardware levels, may exhibit even less physical object transportation disturbance than a consistent hashing algorithm, at least for lower level nodal changes, because the hashing structure is better able to account for the fact that much of the system remains unchanged when only a smaller (lower level) region of the system has changed.

[0023] That is, referring to Fig. 2, if a new mass storage device is added/deleted to/from storage server SS_1_1, note that the storage capacity of the remaining storage servers across level 202 remain unchanged. Therefore, ideally, many if not all mapping changes that result from the change in storage capacity may be kept within storage server SS_1_1. In a sense, a small change at a lower node does not "ripple up" (or hardly ripples up) to the higher level hashing functions, which, in turn, result in minimal changes to the mappings of the nodes that little/no nodal relationship in the system.

[0024] In practice this is represented by little if any change to the weight factors that are applied to the higher level hashing functions in response to the change. Here, if only a single mass storage device is added/deleted to/from storage server SS_1_!, the total storage capacity of rack Rack_1 would hardly change which, in turn, would result in only the slightest of changes, if any, to the weight values that are provided to the highest level hashing function 211. That is, e.g., the change to the percentage of the overall object storage's capacity that Rack_1 represents is almost negligible, and, therefore, the weight values that are provided to the highest hashing level 211 would suffer almost negligible change in response. As such, the mapping determinations made at the highest hashing level 211 would hardly experience any change at all.

[0025] Contra-wise, the weight values provided to the lowest hashing level 213 that resides directly above where the change occurred (the hashing function of server SS_1_1) may see much more significant change (the addition/removal of the storage device to/from server SS_1_1 noticeably changes the server's total storage capacity). As such, there may be more substantial mapping change and corresponding object movement within server SS_1_1 and even less change (but perhaps still noticeable change) within the other servers that are part of Rack_1. That is, object mapping changes are kept more local to the point of the addition/removal disturbance.

[0026] In furtherance of this property, other embodiments may be designed to forcibly prevent weight value changes for hashing instances that are not local to a disturbance. For example, referring to the prior example where server SS_1_1 has a mass storage device added or removed, in an embodiment, the only weight value that is allowed to change is the weight value for storage server SS_1_1 - all other weight values in the system are deliberately kept fixed. By preventing all other weight values from changing, all hashing instances other than the hashing instance of server SS_1_1 will not modify any of their object mappings.

[0027] As such, nearly the entire system does not realize any object mapping changes and corresponding object movements in response to the capacity change. Instead, all the mapping changes and corresponding movements that result from the storage capacity change are forcibly confined within server SS_1_1. Here, for instance, weight values may be kept fixed in response to a capacity change unless some threshold amount of capacity change is experienced in a particular node. For example, if the storage capacity of server SS_1_1 changes by 10% or more, then, other weight values outside of server SS_1_1 are allowed to change (the larger system is permitted to adapt to the change).

[0028] According to one approach, weight value changes are permitted to expand outward from the point of disturbance depending on the magnitude of a storage capacity change. For example, if the storage capacity of server SS_1_1 changes by less than 20%, then, only the weight values of server SS_1_1's hashing function instance are permitted to change. This keeps object relocation within server SS_1_1. By contrast, if the storage capacity of server SS_1_1 changes between 20% and 40%, then, the weight values for Rack_1's and server SS_1_1's hashing instances are permitted to be changed (but weight values for all other hashing instances remain fixed). This keeps object relocation within the servers SS_1_1 through SS_1_N of Rack_1. Further still, if the storage capacity of server SS_1_1 changes by more than 40% then, the weight values for the highest level hashing instance 211, Rack_1's hashing instance and server SS_1_1's hashing instance are permitted to be changed in response. The may result in rack to rack object movements, but the weight change for hashing function 211 will be much less than the weight change for Rack_1's hashing instance or server SS_1_1's hashing instance.

[0029] Keeping weight values fixed in response to a storage capacity change may be particularly useful if the storage capacity change is not only small but temporary. An example is when a mass storage device is being maintained. In this case, the mass storage device may only be temporarily off-line (it can reliably keep its data but cannot process any read or write requests during the maintenance procedure). According to one embodiment, when a mass storage device is being maintained and is temporarily off-line, no weight values are changed in the mass storage device's server or elsewhere in the storage system.

[0030] Here, any read or write requests that are received by the off-line device's server that map to the off-line device are temporarily queued in free (unused) storage capacity of one or more working mass storage devices in the same server as the temporarily off line mass storage device. When maintenance is complete, the read or write requests that have been queued in the server's free storage space are then "replayed" to the mass storage device so that it can service them (it receives the stream of requests that it "missed" while being maintained).

[0031] With respect to the specific type of hashing algorithm, rendezvous hashing algorithms are able to accept weights as described above in order to provide some deliberate imbalance in the hashing output. Thus, in various embodiments, each of the hashing algorithms 211, 212, 213 of Fig. 2 are implemented as a rendezvous hashing algorithm. In various embodiments, the weight values indicate the existence of each immediately lower node and some capacity and/or performance metric for each such immediately lower node.

[0032] Here, although not depicted in Fig. 2 for illustrative convenience, the object storage system may include a centralized or distributed management function implemented on at least one computing system that is communicatively coupled to the storage servers 202 and maintains awareness of their storage capacity levels, changes to their storage capacity levels, their I/O bandwidth capabilities, changes to their bandwidth capabilities, etc. From this awareness the management function crafts the weight values for each hashing level and provides them to the hashing level when appropriate (e.g., in response to a capacity change). The management function may also maintain, e.g., percentage capacity changes and corresponding percentage capacity change thresholds that trigger the sending of a new set of weight values to a hashing function.

[0033] Each hashing function instance may be implemented with dedicated hardware logic circuitry, programmable logic circuitry (e.g., field programmable gate array (FPGA), programmable logic array (PLA), programmable logic device (PLD), etc.) or logic circuitry that executes some form of program code (e.g., a general purpose processor (embedded or otherwise), a special purpose processor (e.g., a digital signal processor (DSP), etc.) or any combination of these. Such circuitry may be located in any of a number of different hardware systems/components of an object storage system such as: 1) a network gateway, networking switch and/or networking router; 2) a storage server; 3) a proxy server or server that acts as a gateway to the object storage system; 4) an I/O interface of a storage server to which multiple mass storage devices attach, etc.

[0034] Any of the storage server and proxy server may be implemented with a computing system. Moreover, elements of networking equipment (gateway/switch/router) may resemble or otherwise include basic components of a computing system. For example, networking equipment typically includes a management control function that includes one or more CPU cores and memory to keep the program code and data that are executed by the CPU cores. Regardless, as described above, the hashing function may be implemented by any of dedicated logic circuitry, programmable logic circuitry, or program code execution or any combination of these within any of a computing system or networking system. The networking equipment may include any of switching logic circuitry and/or routing logic circuitry and ingress/egress ports. A request is received at an ingress port and processed by the switching/routing logic circuitry in accordance with the hash result to direct the request to an appropriate egress port through which the correct next lower level node can be reached.

[0035] Note that the above discussion has essentially focused on a process for determining the storage location of an object based on its name. Generally the object operations include PUT operations (in which new objects are entered into the system and are directed to available storage capacity); GET operations (that operate as read operations in which a read request is directed to the correct storage location to retrieve an object); MODIFY operations (that operate as write operations in which a write request containing new data to be written over an existing object that is stored in the system is directed to the correct storage location of the object); and, DELETE operations (in which a command to delete an object is directed to the correct storage location so that the storage hardware can be instructed to delete the object or permit the object to be overwritten).

[0036] For each of these operations, in various embodiments, a request is directed to the appropriate storage location of an object where the propagation of the request through the system is performed consistently with the multi-level hashing sequence described above (e.g., the request includes the object's name which is hashed at multiple levels of the infrastructure hardware). The different requests, however, have different items attached the request. Specifically, in the case of a PUT, a new object is appended to the request. In the case of a GET no object data need be appended to the request. In the case of a MODIFY, new data for an object (or a new version of the object) is appended to the request. In the case of a DELETE operation, no object data need be appended to the request.

[0037] Although embodiments above have indicated that all nodes have hardware associated with them, in various embodiments nodes may be purely logical and have no hardware characterization about them. In this case, such logical nodes incorporate the weights of the nodes beneath them but do not add to them. In a sense, the weight factors beneath the logical node passes through the logical node to a higher level node.

[0038] In various embodiments, to effect some higher level purpose, administrators may set weight factors that are different than what the analysis of lower nodes would otherwise present, and/or, prevent weigh factors from changing even though there has been a change to the lower nodes.

[0039] Fig. 3 shows a method described above. The method includes performing 301 a first hash on a name of an object with a first hardware element of an object storage system, the name of the object being part of a request that is associated with the object, a result of the first hash identifying a second hardware element directly beneath the first hardware element in a hierarchical arrangement of hardware elements in the object storage system. The method also includes sending 302 the request to the second hardware element. The method also includes performing 303 a second hash on the name of the object with the second hardware element, a result of the second hash identifying a third hardware element directly beneath the second hardware element in the hierarchical arrangement. The method also includes sending 304 the request to the third hardware element to advance the request toward being serviced by the object storage system.

[0040] FIG. 4 provides an exemplary depiction of a computing system 400 (e.g., a smartphone, a tablet computer, a laptop computer, a desktop computer, a server computer, etc.). As observed in FIG. 4, the basic computing system 400 may include a central processing unit 401 (which may include, e.g., a plurality of general purpose processing cores 415_1 through 415_X) and a main memory controller 417 disposed on a multi-core processor or applications processor, system memory 402, a display 403 (e.g., touchscreen, flat-panel), a local wired point-to-point link (e.g., USB) interface 404, various network I/O functions 405 (such as an Ethernet interface and/or cellular modem subsystem), a wireless local area network (e.g., WiFi) interface 406, a wireless point-to-point link (e.g., Bluetooth) interface 407 and a Global Positioning System interface 408, various sensors 409_1 through 409_Y, one or more cameras 410, a battery 411, a power management control unit 412, a speaker and microphone 413 and an audio coder/decoder 414.

[0041] An applications processor or multi-core processor 450 may include one or more general purpose processing cores 415 within its CPU 401, one or more graphical processing units 416, a memory management function 417 (e.g., a memory controller) and an I/O control function 418. The general purpose processing cores 415 typically execute the operating system and application software of the computing system. The graphics processing unit 416 typically executes graphics intensive functions to, e.g., generate graphics information that is presented on the display 403. The memory control function 417 interfaces with the system memory 402 to write/read data to/from system memory 402. Assuming you will add details on persistent memory (e.g. 3DXPoint memory).The power management control unit 412 generally controls the power consumption of the system 400.

[0042] Each of the touchscreen display 403, the communication interfaces 404-407, the GPS interface 408, the sensors 409, the camera(s) 410, and the speaker/microphone codec 413, 414 all can be viewed as various forms of I/O (input and/or output) relative to the overall computing system including, where appropriate, an integrated peripheral device as well (e.g., the one or more cameras 410). Depending on implementation, various ones of these I/O components may be integrated on the applications processor/multi-core processor 450 or may be located off the die or outside the package of the applications processor/multi-core processor 450. The computing system also includes non-volatile storage 420 which may be the mass storage component of the system.

[0043] The processor 450 may also include embedded NVRAM as described above to improve overall operation of various monitoring programs that execute on one or more of the CPU cores 415.

[0044] Embodiments of the invention may include various processes as set forth above. The processes may be embodied in machine-executable instructions. The instructions can be used to cause a general-purpose or special-purpose processor to perform certain processes. Alternatively, these processes may be performed by specific/custom hardware components that contain hardwired logic circuitry or programmable logic circuitry (e.g., field programmable gate array (FPGA), programmable logic device (PLD)) for performing the processes, or by any combination of programmed computer components and custom hardware components.

[0045] Elements of the present invention may also be provided as a machine-readable medium for storing the machine-executable instructions. The machine-readable medium may include, but is not limited to, floppy diskettes, optical disks, CD-ROMs, and magnetooptical disks, FLASH memory, ROMs, RAMs, EPROMs, EEPROMs, magnetic or optical cards, propagation media or other type of media/machine-readable medium suitable for storing electronic instructions. For example, the present invention may be downloaded as a computer program which may be transferred from a remote computer (e.g., a server) to a requesting computer (e.g., a client) by way of data signals embodied in a carrier wave or other propagation medium via a communication link (e.g., a modem or network connection).

[0046] In the foregoing specification, the invention has been described with reference to specific exemplary embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention as set forth in the appended claims. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

Claims

1. An apparatus, comprising
a semiconductor chip comprising logic circuitry to perform a hash on a name of an object of an object storage system, the hash to be performed by a first hardware element in a hierarchical arrangement of hardware elements in an object storage system, the name being part of a request that is associated with the object, a result of the hash to identify a second hardware element directly beneath the first hardware element in the hierarchical arrangement, the request to be sent to the second hardware element to advance the request toward being serviced by the object storage system, the hash incorporating one or more weight factors that describe performance characteristics of hardware elements beneath the hardware element in the hierarchical arrangement of hardware elements.

2. The apparatus of claim 1 wherein the logic circuitry is any combination of:

dedicated hardwired logic circuitry;
programmable logic circuitry;
logic circuitry to execute program code.

3. The apparatus of claim 1 wherein the semiconductor chip is integrated into an object storage system hardware element.

4. The apparatus of claim 2 wherein the object storage system hardware element is any of:

a proxy server;
a networking switch;
a rack gateway;
a storage server;
an I/O interface through which multiple mass storage devices are to be coupled.

5. The apparatus of claim 4 wherein the object storage system hardware element is functionally integrated into the object storage system.

6. A computing system containing the subject matter of any of claims 1 through 5.

7. A machine readable storage medium containing program code that when processed by a processor of a first hardware element in a hierarchical arrangement of hardware elements in an object storage system, causes the processor to implement a method, comprising:
performing a hash on a name of an object of the object storage system, the name being part of a request that is associated with the object, a result of the hash to identify a second hardware element directly beneath the first hardware element in the hierarchical arrangement, the request to be sent to the second hardware element to advance the request toward being serviced by the object storage system.

8. An apparatus, comprising:

means for performing a first hash on a name of an object with a first hardware element of an object storage system, the name of the object being part of a request that is associated with the object, a result of the first hash identifying a second hardware element directly beneath the first hardware element in a hierarchical arrangement of hardware elements in the object storage system;
means for sending the request to the second hardware element; and,
means for performing a second hash on the name of the object with the second hardware element, a result of the second hash identifying a third hardware element directly beneath the second hardware element in the hierarchical arrangement; and,
means for sending the request to the third hardware element to advance the request toward being serviced by the object storage system.

9. The apparatus of claim 8 further comprising means for storing the object in the third hardware element, the third hardware element being a mass storage device.

10. The apparatus of claim 8 wherein one of the second and third hardware elements is a storage server comprising having multiple mass storage devices.

11. The apparatus of claim 8 wherein one of the hardware elements is a networking gateway for a rack of storage servers.

12. The apparatus of claim 8 wherein the first hash incorporates first weight information that describes the structure of the hierarchical arrangement directly beneath the first hardware element.

13. An apparatus, comprising:

means for determining weight factors for leaf nodes of an object storage system's hierarchical arrangements of nodes, the leaf nodes corresponding to storage devices, the weight factors describing performance characteristics of the storage devices;
means for determining weight factors for intermediate nodes above the leaf nodes in the object storage system's hierarchical arrangement of nodes, the weight factors for the intermediate nodes incorporating performance characteristics of respective lower nodes in the object storage system's arrangement of nodes;
means for mapping objects for storage in the object storage system by hashing names of the objects with respective hashing functions positioned at deepening levels of the object storage system's hierarchical arrangement of nodes, the hashing functions incorporating certain ones of the weight factors that describe performance characteristics of respective lower nodes in the object storage system's arrangement of nodes.

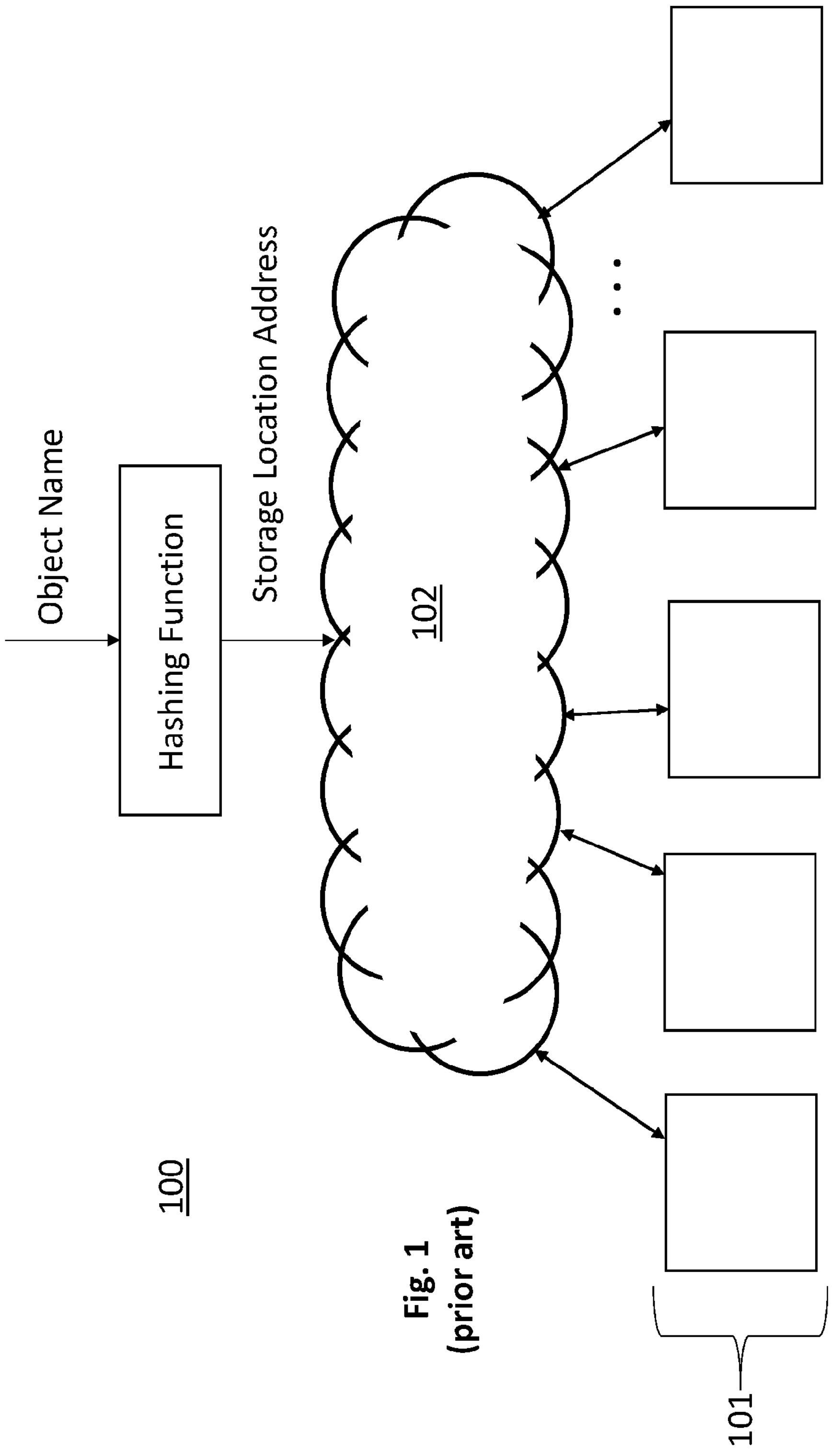
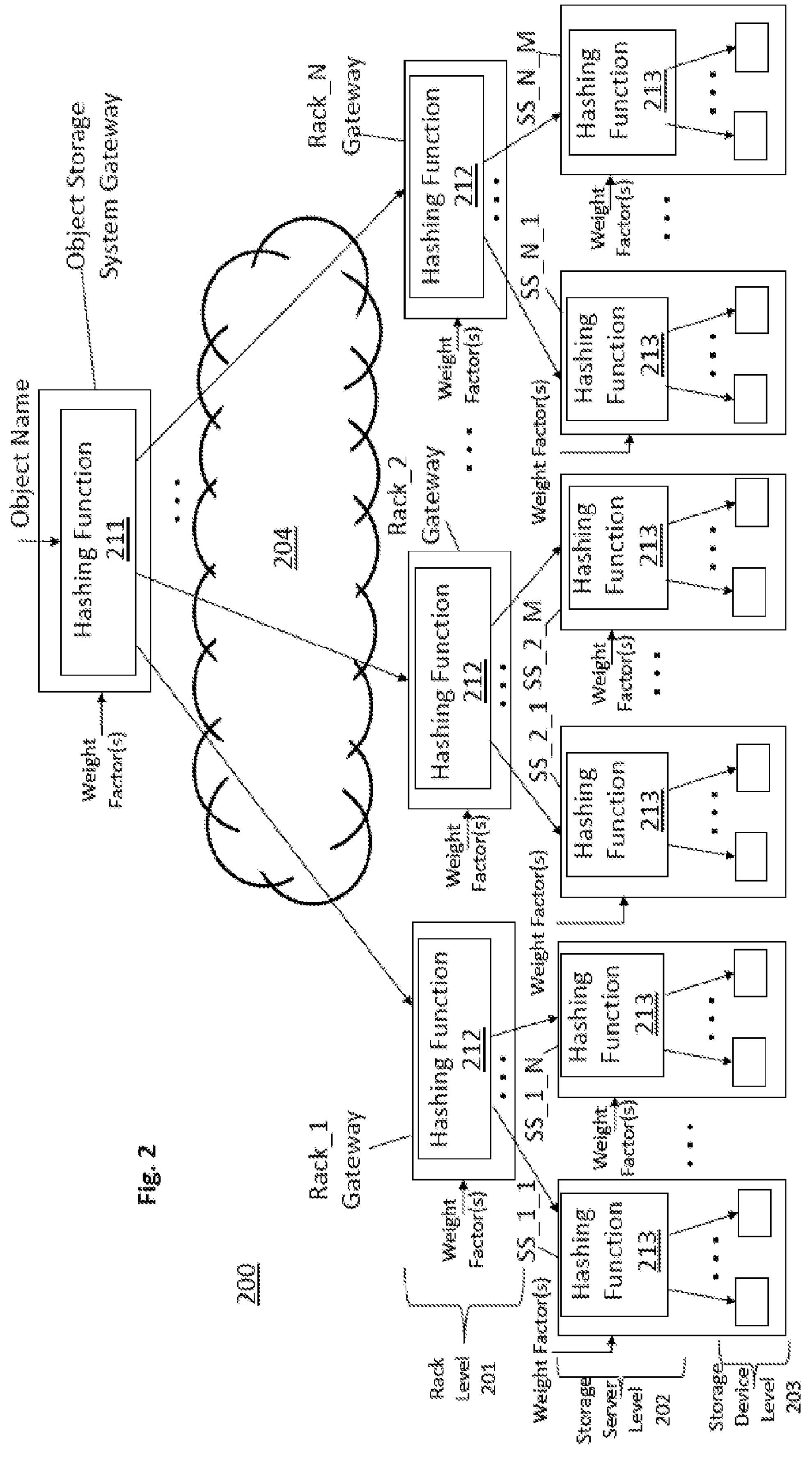


Fig. 1
(prior art)

Fig. 2


200
Object Name
Object Storage System Gateway
Hashing Function 211
Weight Factor(s)
204
Rack_1 Gateway
Rack_2 Gateway
Rack_N Gateway
Hashing Function 212
Weight Factor(s)
SS_1_1
SS_1_N
SS_2_1
SS_2_M
SS_N_1
SS_N_M
Hashing Function 213
Weight Factor(s)
Rack Level 201
Storage Server Level 202
Storage Device Level 203

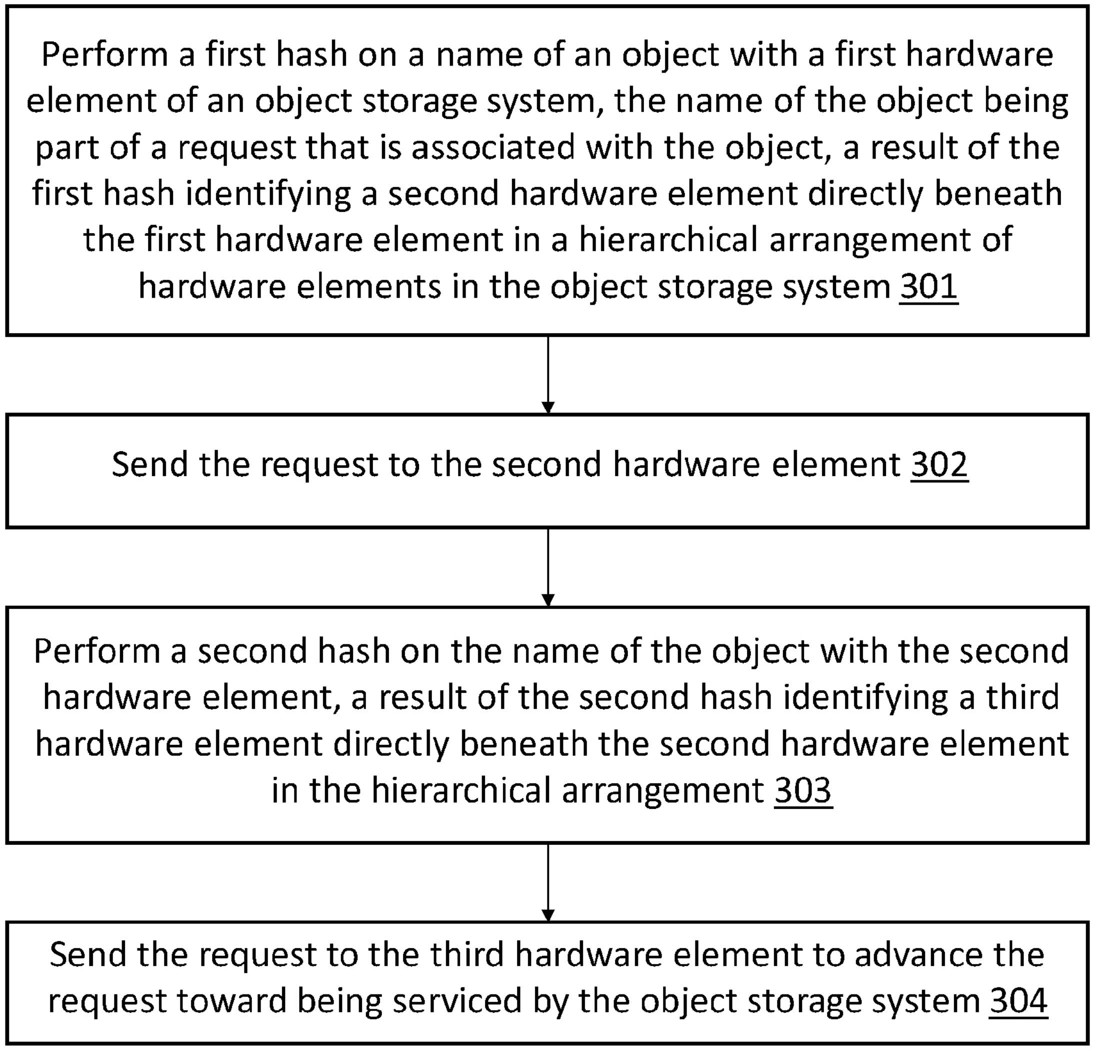

Perform a first hash on a name of an object with a first hardware element of an object storage system, the name of the object being part of a request that is associated with the object, a result of the first hash identifying a second hardware element directly beneath the first hardware element in a hierarchical arrangement of hardware elements in the object storage system 301
Send the request to the second hardware element 302
Perform a second hash on the name of the object with the second hardware element, a result of the second hash identifying a third hardware element directly beneath the second hardware element in the hierarchical arrangement 303
Send the request to the third hardware element to advance the request toward being serviced by the object storage system 304

Fig. 3

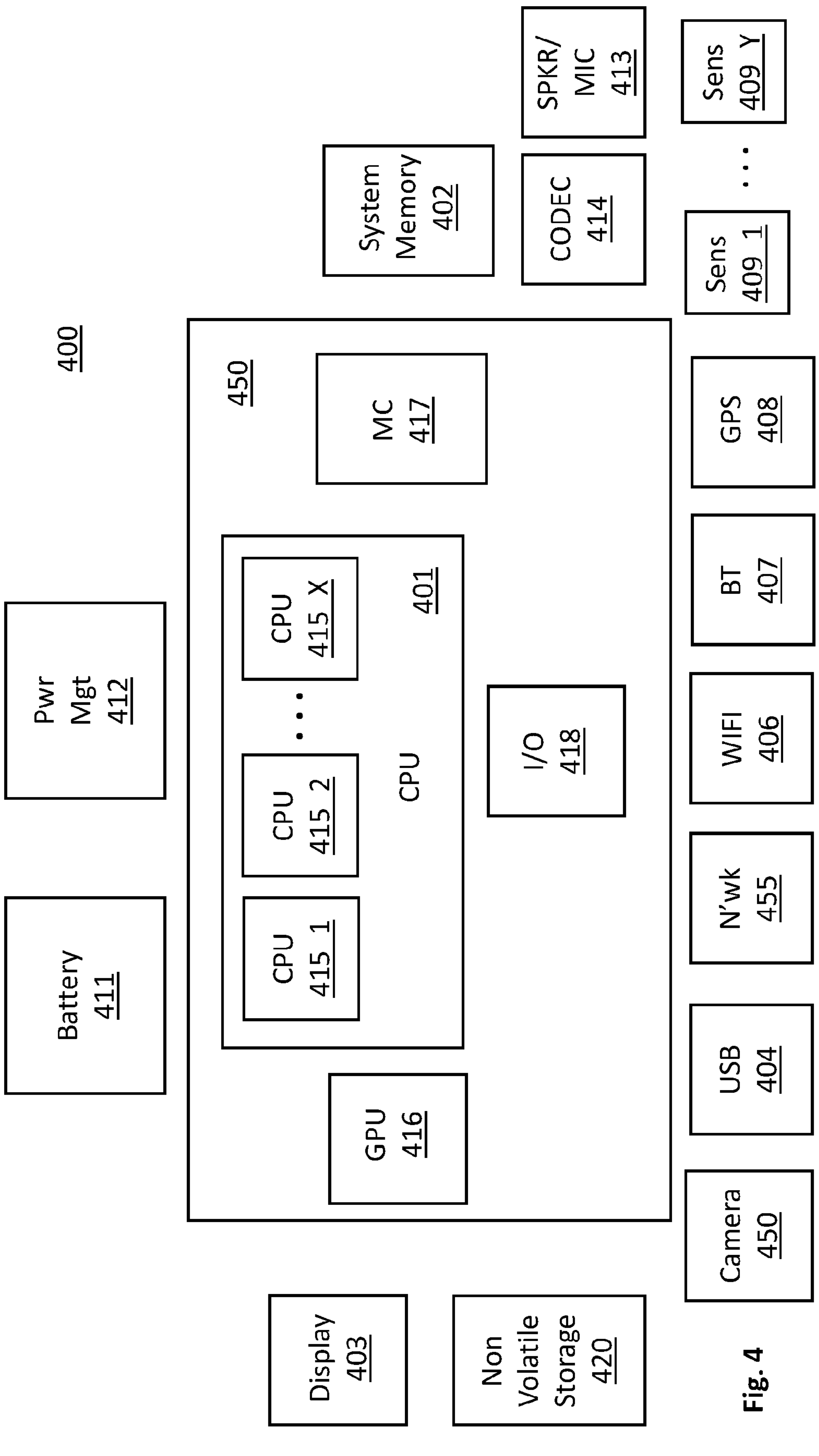

400
Battery 411
Pwr Mgt 412
450
GPU 416
CPU 415_1
CPU 415_2
CPU 415_X
CPU
401
I/O 418
MC 417
System Memory 402
SPKR/MIC 413
CODEC 414
Sens 409_1
Sens 409_Y
Display 403
Non Volatile Storage 420
Camera 450
USB 404
N'wk 455
WIFI 406
BT 407
GPS 408

Fig. 4

Europäisches Patentamt
European Patent Office
Office européen des brevets

EUROPEAN SEARCH REPORT

Application Number
EP 19 15 9540

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (IPC)
X	US 2016/188591 A1 (BESTLER CAITLIN [US] ET AL) 30 June 2016 (2016-06-30) * paragraphs [0003] - [0010], [0066], [0157], [0173] - [0174], [0423] - [0425], [0703] - [0707]; figures 1,3 * -----	1-13	INV. G06F3/06 G06F16/13 G06F12/0864
A	ELGHAMRAWY SALLY M ET AL: "A partitioning framework for Cassandra NoSQL database using Rendezvous hashing", JOURNAL OF SUPERCOMPUTING, KLUWER ACADEMIC PUBLISHERS, DORDRECHT, NL, vol. 73, no. 10, 6 April 2017 (2017-04-06), pages 4444-4465, XP036321672, ISSN: 0920-8542, DOI: 10.1007/S11227-017-2027-5 [retrieved on 2017-04-06] * page 4446, line 1 - line 3 * -----	1-13	
			TECHNICAL FIELDS SEARCHED (IPC)
			G06F

The present search report has been drawn up for all claims

1

Place of search	Date of completion of the search	Examiner
The Hague	27 June 2019	Limacher, Rolf

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 19 15 9540

27-06-2019

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2016188591 A1	30-06-2016	US 2016188591 A1	30-06-2016
		US 2016191250 A1	30-06-2016
		US 2016191508 A1	30-06-2016
		US 2016191509 A1	30-06-2016