

A SELF-SUFFICIENT IN-MEMORY DISTRIBUTED CACHE

BACKGROUND

5 **[0001]** With the recent proliferation in data, organizations continue to take considerable efforts to store and manage the data. End-user applications, increasing network bandwidths, technological developments in communication devices are some of the factors that contribute to the proliferation of data. To support time-critical applications, software systems need to be designed and implemented with high-
10 performance, low-latency processing capabilities, and robust fault-tolerance mechanisms to handle errors and failures. Additionally, such systems need to be thoroughly tested and validated to ensure their reliability and safety in high-stress, time-sensitive environments.

15 **[0002]** In order to achieve the low-latency processing requirement, many applications resort to usage of an in-memory cache. An in-memory cache is a high-speed data storage layer that resides in the random-access memory (RAM) of a computer or server. It is used to temporarily store frequently accessed data and reduce the response time of applications that require rapid access to data. By storing data in memory, it can be accessed and retrieved faster than if it were stored on disk or in a
20 database.

25 **[0003]** Maintaining consistency of a cache in a distributed server cluster is a non-trivial task. A distributed server cluster includes one or more servers that may be deployed in different geographical regions. To ensure that the in-memory cache of each server in such a system is up to date, a centralized cache database is employed.
Each of the servers in the distributed server cluster periodically communicates with the centralized cache (e.g., via a communication network) to update its in-memory cache. There are several disadvantages with such an architecture. For example, the centralized cache architecture is not scalable. Deployment of additional servers in the distributed server cluster results in additional communication channels being setup
30 between the newly deployed servers and the centralized cache for the purposes of enabling the servers to update their respective in-memory caches.

[0004] In such a scenario, the network bandwidth may be exhausted and/or a probability of incurring a bottleneck in the communication network increases. This leads to the system architecture to be unscalable. Furthermore, by utilizing a single caching mechanism (i.e., the centralized cache), runs the risk of providing a stable and highly available caching mechanism. For example, a failure of the caching database and/or a communication link in the communication network may result in one or more servers of the distributed server cluster storing stale data. Additionally, the usage of an external caching component (e.g., a third-party tool/service for caching) makes the overall system prone to errors i.e., increases the risk of failure of the system. Thus, there is a requirement for achieving the same goal of creating a consistent cache across multiple servers without the utilization of any additional infrastructure.

[0005] Embodiments of the invention address these and other problems, individually and collectively.

15

BRIEF SUMMARY

[0006] Embodiments provide for a framework to enable a self-sufficient in-memory distributed cache system without utilizing any external tools or services.

[0007] One embodiment includes a method including: receiving, by a messaging queue module in a messaging computer, a refresh event message corresponding to a data event, the refresh event message indicating that a persistent database is updated with second data corresponding to the data event; and transmitting, by the messaging queue module in the messaging computer, notification messages to a plurality of host servers respectively comprising in-memory caches that stores first data, wherein responsive to receiving the notification messages, the plurality of host servers retrieve second data from the persistent database and update the in-memory caches with the second data.

[0008] Another embodiment includes a messaging computer comprising: a processor; and a computer readable medium, the computer readable medium comprising code, executable by the processor, for implementing a method comprising: receiving, by a messaging queue module in the messaging computer, a refresh event

message corresponding to a data event, the refresh event message indicating that a persistent database is updated with second data corresponding to the data event; and transmitting, by the messaging queue module in the messaging computer, notification messages to a plurality of host servers respectively comprising in-memory caches that
5 stores first data, wherein responsive to receiving the notification messages, the plurality of host servers retrieve second data from the persistent database and update the in-memory caches with the second data.

[0009] Another embodiment includes a method comprising: receiving, by a host server, a notification message from a messaging queue module in a messaging
10 computer, the notification message indicative of a persistent database being updated with second data corresponding to a data event; responsive to receiving the notification message, retrieving by the host server, the second data from the persistent database; and updating, by the host server, an in-memory cache of the host server that stores first data with the second data retrieved from the persistent database.

15 **[0010]** Further details regarding embodiments of the invention can be found in the Detailed Description and the Figures.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] FIG. 1 shows a block diagram of a distributed cache system according
20 to an embodiment.

[0012] FIG. 2 depicts a flow diagram illustrating interactions between different components of the distributed cache system according to some embodiments.

[0013] FIG. 3 shows a block diagram of a host server according to an embodiment.

25 **[0014]** FIG. 4 shows a block diagram of a messaging computer according to an embodiment.

DETAILED DESCRIPTION

[0015] Prior to discussing embodiments of the invention, some terms can be described in further detail.

[0016] A “client device” may be a “mobile device” that may comprise any electronic device that may be transported and operated by a user, which may also provide remote communication capabilities to a network. Examples of remote communication capabilities include using a mobile phone (wireless) network, wireless data network (e.g., 3G, 4G or similar networks), Wi-Fi, Wi-Max, or any other communication medium that may provide access to a network such as the Internet or a private network. Examples of mobile devices include mobile phones (e.g., cellular phones), PDAs, tablet computers, net books, laptop computers, personal music players, hand-held specialized readers, wearable devices (e.g., watches), vehicles (e.g., cars), etc. A mobile device may comprise any suitable hardware and software for performing such functions and may also include multiple devices or components (e.g., when a device has remote access to a network by tethering to another device - i.e., using the other device as a relay - both devices taken together may be considered a single mobile device).

[0017] A “server computer” is typically a powerful computer or cluster of computers. For example, the server computer can be a large mainframe, a minicomputer cluster, or a group of servers functioning as a unit. In one example, the server computer may be a database server coupled to a Web server.

[0018] “Persistent data storage” refers to the ability to store data in a non-volatile manner, meaning the data remains intact even when power is lost, or the system is restarted i.e., persistent data storage is a non-volatile data storage. Persistent data storage may be achieved through various hardware and software solutions, such as hard disk drives (HDDs), solid-state drives (SSDs), optical discs, magnetic tapes, and even cloud storage services. These storage mediums allow for long-term data retention, retrieval, and modification. Data is stored in persistent storage using a file system, which organizes and manages the data in a hierarchical structure, enabling efficient access and management. The file system provides methods for creating, reading, updating, and deleting files or directories. One of the

key characteristics of persistent data storage is durability. It ensures that data remains intact even in the event of hardware failures, power outages, or system crashes.

[0019] An “in-memory cache” is a component or system that stores frequently accessed or computationally expensive data in the main memory (RAM) of a computer or server. It is designed to provide rapid access to data and improve the overall performance of applications by reducing the need to fetch data from slower storage systems, such as disk drives or databases. The basic idea behind an in-memory cache is to keep a copy of data that is likely to be accessed again in the near future in a faster and more accessible location. By storing data in RAM, which has significantly lower latency and higher throughput compared to disk-based storage, the cache can deliver faster response times and improve the overall efficiency of an application. When a requested piece of data is not found in the cache, the cache system fetches the data from the underlying data source (e.g., a database or a remote server) and stores it in the cache for future requests. Subsequent requests for the same data can be served directly from the cache, eliminating the need to access the slower storage system again.

[0020] Event data is information related to an event or operation. For example, if the event is a login operation into an application, the event data can include information such as, but not limited to a time of login, an IP address used to login, etc. Such information can be used for different purposes like tracking user activity and analytics. As another example, in the context of transaction processing e.g., financial transactions, event data may correspond to information including primary account numbers of users, information identifying a user e.g., address of the user, a geographical location of the user, etc.

[0021] Data generators are devices that can detect events and generate data corresponding to the events. Data generators can generate data in various formats, such as structured data (e.g., CSV files, SQL databases), unstructured data (e.g., text documents, images), or even streaming data in real-time.

[0022] A “message queue module” is a communication mechanism used to facilitate exchange of messages between different components/entities within a distributed system. It provides a reliable and scalable way to decouple a sender and a receiver, allowing them to operate independently and at different speeds (if

required). In a message queue, messages are sent by producers (senders) and received by consumers (receivers). In some implementations, the messages are stored in a queue or buffer until they are processed by the consumers. This enables asynchronous communication, where producers can continue generating messages without waiting for immediate processing by the consumers. By aspects of the present disclosure, a message queue may be realized as a message queue module that is included in a messaging computer and configured to process incoming messages e.g., from data generators and transmit notification messages (e.g., providing an indication that one or more datasets in a persistent database has been modified) to one or more host servers.

[0023] A “load balancer” is a networking device or software component that distributes incoming network traffic across multiple servers or resources in a balanced manner. It acts as an intermediary between clients or users and a group of backend servers, ensuring efficient resource utilization, high availability, and optimal performance. The primary purpose of a load balancer is to evenly distribute the incoming workload across multiple servers to prevent any single server from becoming overloaded or overwhelmed. By distributing the traffic, a load balancer can enhance system performance, responsiveness, and scalability. It allows organizations to handle increased traffic and requests without relying on a single server, thereby improving overall reliability and availability.

[0024] An “acquirer” may include an entity that processes payments on behalf of a resource provider, such as a merchant. An acquirer may comprise a financial institution, such as a bank, that maintains an account for a merchant. An acquirer may operate an “acquirer computer,” a computer system that can be used to transmit payment information through networks such as the Internet, including, for example, authorization request messages and authorization response messages.

[0025] “Payment credentials” may include any suitable information associated with an account (e.g., a payment account and/or payment device associated with the account). Such information may be directly related to the account or may be derived from information related to the account. Examples of account information may include a PAN (primary account number or “account number”), user name, expiration date, CVV (card verification value), dCVV (dynamic card verification value), CVV2 (card

verification value 2), CVC3 card verification values, etc. CVV2 is generally understood to be a static verification value associated with a payment device. CVV2 values are generally visible to a user (e.g., a consumer), whereas CVV and dCVV values are typically embedded in memory or authorization request messages and are not readily
5 known to the user (although they are known to the issuer and payment processors). Payment credentials may be any information that identifies or is associated with a payment account. Payment credentials may be provided in order to make a payment from a payment account. Payment credentials can also include a user name, an expiration date, a gift card number or code, and any other suitable information.

10 **[0026]** A “memory” may be any suitable device or devices that can store electronic data. A suitable memory may comprise a non-transitory computer readable medium that stores instructions that can be executed by a processor to implement a desired method. Examples of memories may comprise one or more memory chips, disk drives, etc. Such memories may operate using any suitable electrical, optical,
15 and/or magnetic mode of operation.

[0027] A “processor” may refer to any suitable data computation device or devices. A processor may comprise one or more microprocessors working together to accomplish a desired function. The processor may include CPU comprises at least one high-speed data processor adequate to execute program components for
20 executing user and/or system-generated requests. The CPU may be a microprocessor such as AMD's Athlon, Duron and/or Opteron; IBM and/or Motorola's PowerPC; IBM's and Sony's Cell processor; Intel's Celeron, Itanium, Pentium, Xeon, and/or XScale; and/or the like processor(s).

[0028] A “user” may include an individual. In some embodiments, a user may
25 be associated with one or more personal accounts and/or user devices.

[0029] FIG. 1 depicts a block diagram of a distributed cache system according to an embodiment. The distributed cache system 100 includes a client device 102, an API gateway or load balancer 104, a plurality of host servers 106A-106K, a persistent database 108, a messaging computer 110, and one or more data generators (also
30 referred to herein as data sources) 112A-112M. Each of the host servers 106A-106K comprises an in-memory cache. The host servers 106A-106K may be deployed in different geographical locations.

[0030] The messaging computer 110 includes a message queue module 110A that is configured to receive messages from the data generators 112A-112M. For instance, the message queue module 110A receives a refresh event message corresponding to a data event that is detected/generated by a data generator (e.g.,
5 data generator 112A). The refresh event message indicates to the messaging computer 110 that the persistent database 108 has been updated with information corresponding to the data event (described below). Further, the message queue module 110A is programmed to transmit notification messages to a plurality of host servers 106A-106K in response to the messaging computer 110 receiving the refresh
10 event message. According to some embodiments, a notification message transmitted by the message queue module 110A to the host servers 106A-106K serves the purpose of notifying each of the host servers 106A-106K that their respective in-memory caches may be storing 'stale' data with respect to the data event, and that each of the host servers 106A-106K should obtain 'fresh' data (with respect to the data
15 event) from the persistent database 108. It is appreciated that the operations of reception of the refresh event message and transmission of notification messages by the message computer occur in near real time.

[0031] A data generator (e.g., data generator 112A) is configured to generate/detect a data event and transmit metadata (related to the data event) to be
20 stored in the persistent database 108. The metadata related to the data event may correspond to a dataset e.g., a primary account number (PAN) of a user, identification information of the user, etc., in the case of the data event being a transaction event (e.g., financial transaction). According to some embodiments, the data event is indicative of a modification performed by the data generator with respect to a dataset.
25 Examples of modifications to datasets may include for instance, creation of a new dataset, update an existing dataset, etc. The modified dataset is stored in the persistent database 108. Upon the dataset being successfully stored in the persistent database 108, the data generator generates a refresh event message that is transmitted to the messaging computer 110. It is appreciated that the refresh event
30 message corresponds to the data event and is indicative that the persistent database 108 is updated e.g., storing the modified dataset.

[0032] In one implementation, each of the host servers' 106A-106K registers with the message queue module 110A of the messaging computer 110 to receive

updates for a particular type of event. The message queue module 110A upon receiving the refresh event message, triggers a call back operation to all the participating host servers. In some implementations, the call back operation corresponds to a notification message which indicates to the host servers 106A-106K that a dataset (corresponding to some event) that is currently stored in their in-memory cache (e.g., first data associated with the event) has been updated and that their respective cache is in a stale state. Each of the one or more host servers 106A-106K in response to receiving the notification message from the message queue model, updates their respective in-memory cache by obtaining the updated dataset (e.g., second data corresponding to the data event) from the persistent database 108 and cause their in-memory cache to be in a 'healthy' or 'fresh' state.

[0033] In the distributed caching system 100, a client device 102 can obtain information from the in-memory cache of any one of the host servers' 106A-106K. In some implementations, a request initiated by the client device 102 is transmitted to an API gateway or load balancer 104. The load balancer in turn forwards the request to one of the host servers 106A-106K. As all the host servers are healthy (i.e., in-memory cache is in a healthy state), any one of the host servers 106A-106K can process the request and provide the relevant information to the client device 102. According to some embodiments, the distributed caching system 100 prevents occurrence of replay attacks on the persistent database 108. Replay attacks are defined herein as multiple requests issued by a particular host server and directed to the persistent database 108, for obtaining updated information with respect to a particular data event. In one implementation, the persistent database 108 maintains a mapping e.g., a tuple mapping including a data event and a list of host servers that have already retrieved a fresh copy of data related to the data event. In such a case, the messaging computer 110 may be programmed to re-transmit the notification message corresponding to the data event to only those host servers that have not obtained the fresh copy of data related to the data event. It is appreciated that a particular host server may not have obtained the fresh copy of the data (associated with the data event) in a first attempt due to various factors such as a failure of a communication channel between the host server and the persistent database.

[0034] FIG. 2 depicts a flow diagram 200 illustrating interactions between different components of the distributed cache system 100 according to some

embodiments. The processing depicted in FIG. 2 may be implemented in software (e.g., code, instructions, program) executed by one or more processing units (e.g., processors, cores) of the respective systems, hardware, or combinations thereof. The software may be stored on a non-transitory storage medium (e.g., on a memory device). The method presented in FIG. 2 and described below is intended to be illustrative and non-limiting. Although FIG. 2 depicts the various processing steps occurring in a particular sequence or order, this is not intended to be limiting. In certain alternative embodiments, the steps may be performed in some different order, or some steps may also be performed in parallel.

10 **[0035]** The processing depicted in FIG. 2 commences in step S250, wherein a data generator (e.g., data generator 112A) generates/detects a particular data event and stores information associated with the data event in the persistent database 108. An example of a data event generated by the data generator may correspond to addition of a new customer to a transaction system, and the data corresponding to the
15 data event may correspond to a PAN or BIN of the customer that is newly added to the transaction system.

[0036] In step S252, upon the data generator storing information related to the data event in the persistent database 108, the data generator 112A generates a refresh event message corresponding to a data event. The refresh event message is
20 transmitted from the data generator to the messaging computer 110. The refresh event message may be stored in a message queue module included in the messaging computer 110 and is indicative of the persistent database 108 being updated with data (e.g., fresh data) corresponding to the data event.

[0037] In step S254, the messaging computer 110 upon receiving the refresh event message corresponding to the data event, triggers a call back operation to all
25 the participating host servers 106A-106K (e.g., host servers that have registered with the messaging computer to receive updates of a particular event type). In some implementations, the call back operation corresponds to a notification message that indicates to the host servers 106A-106K, that a dataset (corresponding to the data
30 event) that is currently stored in their in-memory cache (e.g., first data associated with the event) has been updated and that their respective cache is in a stale state.

[0038] Responsive to receiving the notification messages, the host servers 106A-106K in step S256, obtain updated versions of the dataset from the persistent database 108 e.g., second data corresponding to the data event. In step S258, a client device 102 may issue a request to obtain a particular dataset. In some
5 implementations, the request issued by the client device is directed to a load balancer or an API gateway 104. The load balancer 104 may direct the request to any one of the host servers 106A-106K in step S260. The request may be directed by the load balancer 104 to a particular host server based on various criteria e.g., a current load (i.e., number of requests) being handled by the host server, a geographical location of
10 the host server, etc. In step S262, the selected host server (e.g., host server 106A) retrieves the requested information from its in-memory cache and transmits the retrieved information to the client device.

[0039] FIG. 3 depicts a host server 300. By some embodiments, the host server 300 of FIG. 3 corresponds to any one of the host servers 106A-106K included
15 in the distributed cache system 100 depicted in FIG. 1. The host server 300 includes a processor 302, a computer readable medium 304, an in-memory cache 306, and a network interface 308. The host server 300 is also communicatively coupled with a persistent database 108.

[0040] The computer readable medium 304 may comprise a communications
20 module 304A, and an information processing module 304B. The communications module 304A may comprise code that causes the processor 302 to generate messages, forward messages, reformat messages, and/or otherwise communicate with other entities. Specifically, the communication module 304A may include a variety of communications means e.g., short range antennas, long range antennas, etc., to
25 communicate with other devices e.g., the persistent database 108 and/or the client device 102 of FIG. 1.

[0041] By some embodiments, the communications module 304A may include one or more RF transceivers and/or connectors that can be used by host server 300 to communicate with other devices and/or to connect with external networks. The
30 short-range antennas of the communications module 304A may be configured to communicate with external entities through a short-range communication medium (e.g., using Bluetooth, Wi-Fi, infrared, NFC, etc.). The long-range antennas of the

communication module 304A may be configured to communicate with a remote base station and a remote cellular or data network, over the air.

[0042] An instance of a communication channel formed by the host server 300, may be a communication channel formed with the messaging computer 400 (described later with reference to FIG. 4). In such a communication channel, the messaging computer may be programmed to transmit to the host server 300, the refresh event message corresponding to a data event. As another example, a communication channel may be set up between the host server and the persistent database 108. For instance, the host server 300 may setup such a channel to retrieve fresh copies of datasets that are stored in the persistent database. It is appreciated that fresh copies of a dataset may correspond to information such as PANs or BINs of customers that are newly added to a transaction system. As a further example, the host server may also communicate with the client device 102 of FIG. 1 (e.g., via an API gateway or load balancer). In such a communication channel, the client device may be configured to transmit a user identifier associated with a user of the client device. As described herein, the user identifier may be included in a request (issued by the client device 102), wherein the request corresponds to the client device requesting some information (e.g., information stored in the in-cache memory of the host server 300).

[0043] The in-memory cache 306 of the host server is configured to store information that is frequently accessed in a main memory (RAM) of the host server 300. The in-memory cache 306 provides rapid access to data and improves the overall performance of applications by reducing the need to fetch data from slower storage systems, such as the persistent database 108.

[0044] The network interface 308 of the host server may be a network interface card (NIC) or a network adapter i.e., a hardware component that enables the host server 300 to connect and communicate with other devices over a network. The network interface 308 serves as the interface between the host server and the network, thereby allowing data to be transmitted and received. The network interface 308 provides the necessary ports or connectors to establish a network connection. The network interface 308 supports different types of network protocols, such as Ethernet, Wi-Fi, or Fiber Channel, depending on the host server's requirements and

the network infrastructure. The network interface can operate at various speeds, ranging from 10 Mbps (megabits per second) to multiple gigabits per second (Gbps), to accommodate different network bandwidths. In some implementations, when the host server is connected to a network, the network interface 308 facilitates the transmission and reception of data packets. It converts data from the host server into a format suitable for network transmission and vice versa. The network interface may also be configured to handle tasks such as encapsulating data into frames, error checking, and controlling the flow of data.

[0045] The information processing module 304B may be programmed to receive different types of information e.g., a request from the client device 102 of FIG. 1. The information processing module 304B may be configured to process such a request and retrieve from the in-memory cache 306 particular information requested by the client device. Additionally, the information processing module may be configured to process information received from the persistent database 108 and store the processed information in the in-memory cache 306 of the host server. In some implementations, the information processing module 304B may be configured to generate a data packet that includes information such as an identifier of the host server 300. The information processing module 304B may trigger the communications module 304A upon generation of the data packet to transmit the generated data packet to the messaging computer 400. Such a generated data packet may be utilized by the host server 300 for purpose of registering the host server 300 to receive notifications associated with a certain type of data event.

[0046] FIG. 4 shows a block diagram of a messaging computer 400 according to an embodiment. The messaging computer 400 may correspond to the messaging computer 110 included in the distributed cache system 100 depicted in FIG. 1. The messaging computer 400 may comprise a processor 402, which may be coupled to a computer readable medium 404, and a network interface 408. The messaging computer 400 is also communicatively coupled with a persistent database 108. The persistent database 108 may contain mapping information between datasets and host servers. Specifically, for a particular dataset, the persistent database 108 may store a list of host server identifiers corresponding to host servers that have retrieved (and stored) a copy of the dataset in their respective in-memory caches. The computer readable medium 404 may comprise a number of software modules including an

information processing module 404A, a message queue module 404B, and a communications module 404C.

[0047] The communications module 404C may comprise code that causes the processor 402 to generate messages, forward messages, reformat messages, and/or otherwise communicate with other entities. Specifically, the communication module 404C may include a variety of communications means e.g., short range antennas, long range antennas, etc., to communicate with other devices e.g., the persistent database 108 and/or the plurality of host servers 106A-106K of FIG. 1.

[0048] By some embodiments, the communications module 404C may include one or more RF transceivers and/or connectors that can be used by the messaging computer 400 to communicate with other devices and/or to connect with external networks. The short-range antennas of the communications module 404C may be configured to communicate with external entities through a short-range communication medium (e.g., using Bluetooth, Wi-Fi, infrared, NFC, etc.). The long-range antennas of the communication module 304A may be configured to communicate with a remote base station and a remote cellular or data network, over the air. An instance of a communication channel formed by the messaging computer 400, may be a communication channel formed with the host server 300 of FIG. 3. In such a communication channel, the messaging computer may be programmed to transmit to the host server 300, notification messages corresponding to a data event. As another example, a communication channel may be set up between the messaging computer and the one or more data generators (e.g., data generator 112A of FIG. 1). Such a communication channel may be utilized by the messaging computer to receive, from the data generator, a refresh event message corresponding to a data event. It is noted that the refresh event message indicates that the persistent database 108 is updated with data corresponding to the data event. In some implementations, the messaging computer may also establish a communication channel with the persistent database 108 to retrieve a list of identifiers of the plurality of host servers that have updated their respective in-memory caches with respect to a particular dataset. Upon obtaining such a list of host servers, the messaging computer in one embodiment, may be configured to retransmit the notification message only to those host servers that have not updated their in-memory caches with the updated data. In such a manner, the distributed cache system 100 of FIG. 1 prevents replay attacks i.e., one or more

host servers sending multiple requests to the persistent database to update their respective in-memory caches.

[0049] The network interface 408 of the messaging computer may be a network interface card (NIC) or a network adapter i.e., a hardware component that enables the messaging computer to connect and communicate with other devices over a network. The network interface 408 serves as the interface between the messaging computer and the network, thereby allowing data to be transmitted and received. The network interface 408 provides the necessary ports or connectors to establish a network connection. The network interface 408 supports different types of network protocols, such as Ethernet, Wi-Fi, or Fiber Channel, depending on the host server's requirements and the network infrastructure. The network interface can operate at various speeds, ranging from 10 Mbps (megabits per second) to multiple gigabits per second (Gbps), to accommodate different network bandwidths. In some implementations, when the messaging computer is connected to a network, the network interface 408 facilitates the transmission and reception of data packets. It converts data from the messaging computer into a format suitable for network transmission and vice versa. The network interface may also be configured to handle tasks such as encapsulating data into frames, error checking, and controlling the flow of data.

[0050] The information processing module 404A may be programmed to receive different types of information e.g., requests from different host servers pertaining to registering the host servers for being notified with respect to certain types of data events. The information processing module 404A may be configured to process such requests and maintain a list of registered host servers and associated data event types in its memory.

[0051] Additionally, the information processing module 404A may be configured to process information received from a data generator. For instance, the information processing module 404A may receive a refresh event message (from the data generator) corresponding to a data event, where the refresh event message indicates that the persistent database is updated with respect to a data event. In turn, the information processing module 404A may be configured to generate a message (e.g., notification message) in response to receiving the refresh event message. The

notification message may be transmitted by the communication module 404C of the messaging computer to a plurality of host servers (e.g., host server 106A), where the notification message serves as an indicator to the host servers that information associated with the data event has been updated and stored in the persistent database

5 108. As stated previously, each of the host servers in turn, communicate with the persistent database to retrieve the updated information.

[0052] The message queue module 404B of the messaging computer 400 is a buffer or on a storage medium where messages (e.g., refresh event messages received from data generators) are stored. In some implementations, responsive to a

10 first retrieval message being stored in the message queue module 404B, the messaging computer is configured to transmit notification messages to the plurality of host servers to indicate that the persistent database is updated with data corresponding to the data event. In some implementations, the operation of receiving the refresh event message and the operation of transmitting the notification message

15 occur in near real-time.

[0053] Embodiments of the invention have several advantages. Embodiments of the invention provide for a mechanism to ensure consistent in-memory caches across a distributed cluster of server computers. Such a distributed architecture is self-sufficient and does not depend on any specialized hardware or software to store data

20 in-memory to provide low latency response. Further, the proposed architecture for managing in-memory caches of the host servers is scalable in nature i.e., to extend the number of host servers, one only has to register the host server with the messaging computer (e.g., with the message queue module) to receive updates with respect to different types of events. The distributed cache system of FIG. 1 may be utilized in a

25 transaction processing system to process different transactions. For instance, the client device 102 may correspond to an acquirer computer and the host servers 106A-106K may correspond to application servers that are geographically distributed and maintain customer information e.g., customer PAN information, customer credential information, other customer identification data etc. In such a setting, the application

30 servers can process requests received from the acquirer computer in a latency-sensitive manner and provide the most up-to date customer information to the acquirer computer for processing different transactions.

[0054] Any of the software components or functions described in this application may be implemented as software code to be executed by a processor using any suitable computer language such as, for example, Java, C, C++, C#, Objective-C, Swift, or scripting language such as Perl or Python using, for example, conventional or object-oriented techniques. The software code may be stored as a series of instructions or commands on a computer readable medium for storage and/or transmission, suitable media include random access memory (RAM), a read only memory (ROM), a magnetic medium such as a hard-drive or a floppy disk, or an optical medium such as a compact disk (CD) or DVD (digital versatile disk), flash memory, and the like. The computer readable medium may be any combination of such storage or transmission devices.

[0055] Such programs may also be encoded and transmitted using carrier signals adapted for transmission via wired, optical, and/or wireless networks conforming to a variety of protocols, including the Internet. As such, a computer readable medium according to an embodiment of the present invention may be created using a data signal encoded with such programs. Computer readable media encoded with the program code may be packaged with a compatible device or provided separately from other devices (e.g., via Internet download). Any such computer readable medium may reside on or within a single computer product (e.g., a hard drive, a CD, or an entire computer system), and may be present on or within different computer products within a system or network. A computer system may include a monitor, printer, or other suitable display for providing any of the results mentioned herein to a user.

[0056] The above description is illustrative and is not restrictive. Many variations of the invention will become apparent to those skilled in the art upon review of the disclosure. The scope of the invention should, therefore, be determined not with reference to the above description, but instead should be determined with reference to the pending claims along with their full scope or equivalents.

[0057] One or more features from any embodiment may be combined with one or more features of any other embodiment without departing from the scope of the invention.

[0058] As used herein, the use of "a," "an," or "the" is intended to mean "at least one," unless specifically indicated to the contrary.

WHAT IS CLAIMED IS:

- 1 1. A method comprising:
2 receiving, by a messaging queue module in a messaging computer, a refresh
3 event message corresponding to a data event, the refresh event message indicating
4 that a persistent database is updated with second data corresponding to the data
5 event; and
6 transmitting, by the messaging queue module in the messaging computer,
7 notification messages to a plurality of host servers respectively comprising in-memory
8 caches that stores first data, wherein responsive to receiving the notification
9 messages, the plurality of host servers retrieve second data from the persistent
10 database and update the in-memory caches with the second data.
11
12 2. The method of claim 1, wherein a first host server included in the
13 plurality of host servers transmits information included in an in-memory cache of the
14 first host server to a client device responsive to receiving a request from the client
15 device.
16
17 3. The method of claim 2, wherein the client device communicates with
18 the plurality of host servers via an API gateway or load balancer.
19
20 4. The method of claim 1, wherein each of the plurality of host servers,
21 registers with the messaging queue module to receive notifications associated with a
22 certain type of data event.
23
24 5. The method of claim 1, wherein the receiving and the transmitting
25 occur in near real time.
26
27 6. The method of claim 1, wherein the persistent database is a non-
28 volatile data storage.
29
30 7. The method of claim 1, wherein the data event corresponds to a
31 transaction event, and the second data comprises a dataset of one or more primary
32 account numbers.

33

34 8. A messaging computer comprising:

35 a processor; and

36 a computer readable medium, the computer readable medium comprising

37 code, executable by the processor, for implementing a method comprising:

38 receiving, by a messaging queue module in the messaging computer, a

39 refresh event message corresponding to a data event, the refresh event message

40 indicating that a persistent database is updated with second data corresponding to

41 the data event; and

42 transmitting, by the messaging queue module in the messaging

43 computer, notification messages to a plurality of host servers respectively comprising

44 in-memory caches that stores first data, wherein responsive to receiving the

45 notification messages, the plurality of host servers retrieve second data from the

46 persistent database and update the in-memory caches with the second data.

47

48 9. The messaging computer of claim 8, wherein a first host server

49 included in the plurality of host servers transmits information included in an in-

50 memory cache of the first host server to a client device responsive to receiving a

51 request from the client device.

52

53 10. The messaging computer of claim 9, wherein the client device

54 communicates with the plurality of host servers via an API gateway or load balancer.

55

56 11. The messaging computer of claim 8, wherein each of the plurality of

57 host servers, registers with the messaging queue module to receive notifications

58 associated with a certain type of data event.

59

60 12. The messaging computer of claim 8, wherein the receiving and the

61 transmitting occur in near real time.

62

63 13. The messaging computer of claim 8, wherein the persistent database is

64 a non-volatile data storage.

65

- 66 14. A method comprising:
67 receiving, by a host server, a notification message from a messaging queue
68 module in a messaging computer, the notification message indicative of a persistent
69 database being updated with second data corresponding to a data event;
70 responsive to receiving the notification message, retrieving by the host server,
71 the second data from the persistent database; and
72 updating, by the host server, an in-memory cache of the host server that
73 stores first data with the second data retrieved from the persistent database.
74
- 75 15. The method of claim 14, further comprising:
76 transmitting, by the host server, information included in the in-memory cache
77 of the host server to a client device in response to receiving a request from the client
78 device.
79
- 80 16. The method of claim 15, wherein the client device communicates with
81 the host server via an API gateway or load balancer.
82
- 83 17. The method of claim 14, further comprising:
84 registering, by the host server, with the messaging queue module included in
85 the messaging computer, wherein the registering causes the host server to receive
86 notifications associated with a certain type of data event.
87
- 88 18. The method of claim 14, wherein the host server receives the
89 notification message from the messaging queue module in the messaging computer,
90 in response to the messaging computer receiving a refresh event message
91 corresponding to the data event from a data generator.
92
- 93 19. The method of claim 14, further comprising:
94 transmitting, by the host server, a request to the persistent database, the
95 request including an identifier of the host server and requesting the second data
96 corresponding to the data event, wherein the persistent database is configured to
97 store the identifier of the host server with respect to the data event in response to the
98 host server completing the updating.

99

100 20. The method of claim 14, wherein the persistent database is a non-
101 volatile data storage.