SYSTEM AND METHOD FOR SELF-CORRECTING CACHE

The present invention relates to a system and method for a self-correcting cache. More particularly but not exclusively it relates to a system and method for self-correcting cache to be used in e-commerce or similar online service applications.

BACKGROUND OF THE INVENTION

The surge in the accessibility of the internet in recent years has been accompanied by a proliferation of online platforms providing various services to users. As network infrastructure improves and bandwidth increases, users expect such online services to have faster loading speeds and present information with a high degree of clarity and accuracy. Tolerance for a slow or confusing user experience is dropping and may even frustrate users to the extent that they stop using a particular service. These aspects of user experience are therefore priorities for the providers of online services when they design their software and network infrastructure.

The servers that form the backbone of online platforms commonly have databases using hard-disks to store the data that users want access to, but due to their mechanical nature hard-disks can have a delay associated with reading data from them. This read delay can manifest as slow response times to users of the online platform when they send requests. For this reason, caches are commonly used to improve loading times for users, where a working copy of some subset of the data is stored in limited, temporary but high-speed Random Access Memory (RAM). Caches can be used on the server side to increase the speed of response if requested data is already in the cache, or on the client side so that the same request does not need to be sent repeatedly. If it is possible for the server's databases to be updated during the use of the online platform, some method of updating caches must be implemented so that users are not being served old data for a prolonged period.

Modern online platforms are commonly developed with a microservices architecture, where the server-side operations are decomposed into a number of microservices that perform tasks independently but all communicate and contribute to the online platform's user interface. Due to the large volume of server-side network communication that occurs when many microservices are used, packet loss and asynchronous timing can lead to cache anomalies when the server's databases are updated with the latest data. These anomalies can cause users to be served inconsistent and confusing data. Correction of anomalies across caches in a microservices architecture poses a challenge that may not be efficiently addressed by conventional methods of

updating caches. It is thus desirable to find systems and methods to detect and rapidly correct these anomalies such that impact on user experience is minimised.

OBJECT OF THE INVENTION

It is an object of the present invention to provide a system and/or method for a self-correcting cache which overcomes or at least partially ameliorates some of the abovementioned disadvantages or which at least provides the public with a useful choice.

Alternately or additionally, it is an object of the present invention to provide a computing system or method which overcomes or at least partially ameliorates some of the abovementioned disadvantages or which at least provides the public with a useful choice.

STATEMENTS OF INVENTION

In a first aspect the present invention may be said to reside in a computing system comprising:

a self-correcting cache engine comprising:

a message system configured to receive a stream of messages from at least one source, store the messages and send the messages to at least one destination, and

a processor implementing or running a composite service, the processor being in operative communication with the at least one source, and the message system, the composite service being configured to:

fetch and process at least one data set from the at least one source and store the at least one data set in a composite service cache that is part of or is operatively connected to the processor in a processed form;

receive the messages sent from the message system, use at least one content of the messages to fetch at least one updated data set from the at least one source, and use the at least one updated data set to update the at least one data set stored in the composite service cache; and

send the processed form of the at least one data set from the composite service cache to a user interface,

wherein the user interface is part of the computing system and is configured to receive and display to a user the processed form of the at least one data set sent from the composite service.

In one embodiment, the at least one source is at least one of:

a message source (that is the source of the messages), and

a data source that is the source of the at least one data set.

In one embodiment, the processed form of the at least one data set from the composite service cache is sent over a network.

In one embodiment, the self-correcting cache engine further comprises at least one processor that is an error detection processor, the at least one error detection processor being in operative communication with the user interface and the message system and implementing an error detector that is configured to:

detect inconsistencies in the processed form of the at least one data set displayed by the user interface; and

generate and send to the message system a correction message whenever an inconsistency is detected, wherein contents of the correction message is information about the at least one data set within which the inconsistency has been detected.

In one embodiment, the error detector is further configured to detect the inconsistencies through analysis of at least one log data set, wherein the at least one log data set is collected during operation of the user interface and is stored in a log data store that is part of or is operatively connected to the at least one error detection processor.

In one embodiment, the one or more data sources include at least one processor that is a data source processor, the at least one data source processor implementing a plurality of microservices, wherein the microservices are configured to process and store some or all of the at least one data sets fetched by the composite service.

In one embodiment, one of the at least one data source processors implements a centralised repository and is operatively connected to a database configured to store some or all of the at least one data set fetched by the composite service.

In one embodiment, the plurality of microservices other than the centralised repository are dependent microservices configured to be dependent on the centralised repository for receiving and storing the at least one data set.

In one embodiment, the at least one data set is intermittently fetched by the composite service.

In one embodiment, some of the plurality of microservices are configured to store data in a plurality of microservice caches.

In one embodiment, the centralised repository is one of the at least one message sources of the message system and is configured to generate an event message whenever the database is updated, wherein contents of the event message are information about the at least one updated data set.

In one embodiment, the at least one data set stored in the composite service cache is configured to expire after a set period of time, thereby triggering the composite service to re-fetch and re-process the at least one data set from the one or more data sources.

In one embodiment, the centralised repository is configured to be updated by one or more external data providers.

In one embodiment, the computing system is an e-commerce platform.

In one embodiment, the one or more external data providers are retail vendors.

In one embodiment, the at least one data set relates to inventory and pricing information for products available for sale.

In one embodiment the message system comprises one or more First In First Out (FIFO) queue structures.

In one embodiment the one or more FIFO queue structures are each associated with one of the plurality of microservices or the composite service, and are configured to store the messages until they are read at all destinations.

In one embodiment the composite service cache is implemented with a distributed Random Access Memory (RAM) storage system.

In one embodiment the microservice caches are implemented with a distributed RAM storage system.

In one embodiment the message system is implemented with a distributed message queue platform.

In a second aspect the present invention may be said to reside in a cache error correction system for a microservices architecture online platform comprising:

a processor implementing an error detector, the error detector configured to:

collect at least one log data set from a user interface of the online platform and store it in a log data store;

continuously analyse the at least one log data set stored in the log data store to identify one or more inconsistencies;

generate and send a correction message whenever the one or more inconsistencies are identified, wherein contents of the correction message are information about the at least one data set that is inconsistent.

In one embodiment the correction message is received by a message system of the online platform.

In one embodiment the message system sends the correction message to a secondary processor implementing a composite service, the composite service configured to:

store at least one data set received from one or more data sources in a composite service cache; and

receive the correction message from the message system, use the contents of the correction message to fetch at least one updated data set from at least one of the one or more data sources, and use the at least one updated data set to update the at least one data set stored in the composite service cache.

In one embodiment, the online platform is an e-commerce platform.

In one embodiment, the at least one log data set relates to products viewed through the user interface.

In one embodiment, the analysis of the at least one log data set involves matching values that correspond to a certain product.

In a third aspect the present invention may be said to reside in a method of cache error correction for an online platform comprising at least the following steps:

collecting at least one log data set as a user navigates the online platform and storing it in a log data store;

continuously analysing the at least one log data set to match associated values and determine if the associated values are inconsistent;

identifying at least one data source that provides at least one data set associated with the inconsistency; and

sending one or more correction messages indicating at least one cache of the online platform to refresh the at least one data set corresponding to the inconsistency.

In one embodiment, the online platform is an e-commerce platform.

In one embodiment, the at least one log data set relates to products viewed by the user.

In one embodiment, analysis of the at least one log data set involves matching values that correspond to a certain product.

In a fourth aspect the present invention may be said to reside in a self-correcting cache engine comprising:

a message system configured to receive a stream of messages from at least one source, store the messages and send the messages to at least one destination; and

a processor implementing or running a composite service, the processor being in operative communication with the at least one source, and the message system, the composite service being configured to:

fetch and process at least one data set from the at least one source and store the at least one data set in a composite service cache that is part of or is operatively connected to the processor in a processed form;

receive the messages sent from the message system, use at least one content of the messages to fetch at least one updated data set from the

at least one source, and use the at least one updated data set to update the at least one data set stored in the composite service cache; and

send the processed form of the at least one data set from the composite service cache to a user interface.

In one embodiment, the at least one source is at least one of:

a message source that is the source of the messages, and

a data source that is the source of the at least one data set.

In one embodiment, the processed form of the at least one data set from the composite service cache is sent over the network.

In one embodiment, the at least one data set is intermittently fetched by the composite service.

In one embodiment, the at least one source is at least one of:

a message source that is the source of the messages, and

a data source that is the source of the at least one data set.

In a fifth aspect the present invention may be said to reside in a method of cache error correction for an online platform comprising at least the following steps:

using a message system to receive a stream of messages from at least one source, store the messages and send the messages to at least one destination;

implementing or running a composite service in a processor that is in operative communication with the at least one source and the message system;

fetching and processing at least one data set from the at least one source and storing the at least one data set in a composite service cache that is part of or is operatively connected to the processor in a processed form;

receiving the messages sent from the message system, using at least one content of the messages to fetch at least one updated data set from the at least one source, and using the at least one updated data set to update the at least one data set stored in the composite service cache;

sending the processed form of the at least one data set from the composite service cache; and

using a user interface to receive and display to a user the processed form of the at least one data set sent from the composite service.

In one embodiment, the method further comprises using at least one error detection processor in operative communication with the user interface and the message system to implement an error detector.

In one embodiment, the method further comprises using the error detector to:

detect inconsistencies in the processed form of the at least one data set displayed by the user interface; and

generate and send to the message system a correction message whenever an inconsistency is detected, wherein contents of the correction message is information about the at least one data set within which the inconsistency has been detected.

In one embodiment, the method further comprises using the error detector to detect the inconsistencies through analysis of at least one log data set, wherein the at least one log data set is collected during operation of the user interface and is stored in a log data store that is part of or is operatively connected to the at least one error detection processor.

In one embodiment, the at least one data set is intermittently fetched by the composite service.

In one embodiment, the method further comprises using the database which is one of the at least one message sources of the message system to generate an event message whenever the database is updated, wherein contents of the event message are information about the at least one updated data set.

In one embodiment, using one data source processor to implement a plurality of microservices.

In one embodiment, the method further comprises using the plurality of microservices to process and store some or all of the at least one data sets fetched by the composite service.

In one embodiment, the database is a centralised repository and the method further comprises using the plurality of microservices to be dependent on the centralised repository for receiving and storing the at least one data set.

In one embodiment, the method further comprises using the plurality of microservices to store data in a plurality of microservice caches.

In one embodiment, the method further comprises triggering the composite service to re-fetch and re-process the at least one data set from the one or more data sources when the at least one data set stored in the composite service cache is expires after a set period of time.

In one embodiment, the method further comprises using the centralised repository to be updated by one or more external data providers.

In one embodiment, the computing system is an e-commerce platform.

In one embodiment, the one or more external data providers are retail vendors.

In one embodiment, the at least one data set relates to inventory and pricing information for products available for sale.

In one embodiment, the message system comprises one or more First In First Out (FIFO) queue structures.

In one embodiment, the method further comprises associating each of the one or more FIFO queue structures with one of the plurality of microservices or the composite service, and are configured to store the messages until they are read at all destinations.

In one embodiment, the composite service cache is implemented with a distributed Random Access Memory (RAM) storage system.

In one embodiment, the method further comprises implementing the microservice caches with a distributed RAM storage system.

In one embodiment, the method further comprises implementing the message system with a distributed message queue platform.

In one embodiment, the method is implemented using the system as defined in the first aspect.

In this specification where reference has been made to patent specifications, other external documents, or other sources of information, this is generally for the purpose of providing a context for discussing the features of the invention. Unless specifically stated otherwise, reference to such external documents is not to be construed as an admission that such documents, or such sources of information, in any jurisdiction, are prior art, or form part of the common general knowledge in the art.

Other aspects of the invention may become apparent from the following description which is given by way of example only and with reference to the accompanying drawings.

As used herein the term "and/or" means "and" or "or", or both.

As used herein "(s)" following a noun means the plural and/or singular forms of the noun.

For purposes of the description hereinafter, the terms "upper", "lower", "right", "left", "vertical", "horizontal", "top", "bottom", "lateral", "longitudinal" and derivatives thereof shall relate to the invention as it is oriented in the drawing figures. However, it is to be understood that the invention may assume various alternative variations, except where expressly specified to the contrary. It is also to be understood that the specific devices illustrated in the attached drawings and described in the following description are simply exemplary embodiments of the invention. Hence, specific dimensions and other physical characteristics related to the embodiments disclosed herein are not to be considered as limiting.

It is acknowledged that the term ''comprise'' may, under varying jurisdictions, be attributed with either an exclusive or an inclusive meaning. For the purpose of this specification, and unless otherwise noted, the term 'comprise' shall have an inclusive meaning, allowing for inclusion of not only the listed components or elements, but also other non-specified components or elements. The terms 'comprises' or 'comprised' or 'comprising' have a similar meaning when used in relation to the system or to one or more steps in a method or process.

As used hereinbefore and hereinafter, "(s)" following a noun means the plural and/or singular forms of the noun.

When used in the claims and unless stated otherwise, the word 'for' is to be interpreted to mean only 'suitable for', and not for example, specifically 'adapted' or 'configured' for the purpose that is stated.

The entire disclosures of all applications, patents and publications, cited above and below, if any, are hereby incorporated by reference.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will now be described by way of example only and with reference to the drawings in which:

Figure 1: shows a basic architecture for a system according to one preferred embodiment of the invention;

Figure 2: shows a microservices architecture for the system of Figure 1;

Figure 3: shows a further microservices architecture for the system of Figure 1 having a centralised repository and caches;

Figure 4: shows a microservices architecture for the system of Figure 1 with cache error correction;

Figure 5: shows a cache error correction flowchart of the system of Figure 1;

Figure 6: is a schematic representation of main elements of the system of Figure 1.

DETAILED DESCRIPTION

With reference to the above drawings, in which similar features are generally indicated by similar numerals, a system for self-correcting cache according to a first aspect of the invention is generally indicated by the numeral 1.

In one embodiment of the system 1, changes to a database 10 are made frequently by external data providers such as vendors 2 and must be quickly reflected in data that is served to users 3. The system 1 may be used in an online platform which may be an e-commerce platform where the external data providers such as vendors 2 provide inventory and pricing data for integration into a single online marketplace. This basic system architecture is shown in Figure 1. As shown, the pricing and inventory information of vendors is stored in the database 10 but could be changed by vendors at any time. A microservices architecture is common for e-commerce and other online platforms, meaning that the user interface 40 is fed by multiple microservices 20 that may have their own databases 10, as shown in Figure 2, or that may all be dependent on database that is a centralised repository 11, as shown in Figure 3. A composite service 30, which may equivalently be known as an orchestrator as in Figure 5, may also be used to compile data from at least some of the microservices 20 for feeding to the user interface 40. In embodiments with a centralised repository 11, the microservices 20 and composite service 30 may have their own microservice caches 21 and composite service cache 31 to store data for fast access. When a change is made to the database 10 to reflect the latest vendor data, a window is created in which users could be served incorrect pricing or inventory information from a cache that has yet to receive the updated data. The “cache accuracy” is an important part of user experience, as users want to know the most up-to-date information. Cache accuracy can be improved through assigning data “Time to Live” (TTL) after which it expires and is either purged from the cache or retrieved again from

database 10, or by having the centralised repository 11 send messages to dependent caches whenever it receives updates.

In a microservices architecture, an additional problem may arise of having to maintain the accuracy and consistency of multiple caches at once so that the composite service 30 that serves the user interface 40 does not send information that has obvious errors. As an example, an e-commerce platform may have a number of interfaces, such as a Search page and a Product Details page, where each one relies on different microservices 20 that are storing some of the same data in a way that provides the fastest access for the specific task. Because some of this data may be replicated between microservice caches 21, inconsistencies may arise between them due to packet loss or asynchronous timing differences in the processing of updated data. A composite service 30 is responsible for compiling and caching data from these multiple microservices 20 as pages ready to display to a user 3, but it cannot verify consistency between any data that is replicated. A case might then arise where a user 3 is served pages that have obvious contradictions between pricing information for example, causing a negative user experience that may be more apparent than other types of cache inaccuracy. It is thus desirable to find ways to maintain cache accuracy and actively correct inconsistencies between replicated data in a microservices architecture, while also keeping response time to a minimum.

In one embodiment now described, there is provided a self-correcting cache engine 4 for system 1. Preferably the system uses the microservices architecture such that it is composed of a number of independent microservices 20 and one or more composite services 30 that collate data so that it can be served to a user interface 40 in a display format. A user 3 may access the online platform from any device provided with network connectivity, which may include a personal computer or a mobile phone. The device may display the user interface 40 of the online platform 1 through a web browser or from a dedicated application installed on the device. In a preferred embodiment, the online platform 1 is an e-commerce platform for which the data is inventory and pricing data provided by one or more external data providers being vendors 2, such that the online platform allows users to view and/or search a variety of products for sale, create accounts or submit shipping information, and complete transactions in order to purchase products. The online platform may be operated by a single vendor, or it may collate products from multiple vendors into a single marketplace.

In one embodiment as shown in Figure 4, the self-correcting cache engine 4 comprises one or more databases 10, a plurality of microservices 20, a composite service 30 having a composite service cache 31, a user interface 40, a message system 50 and an error detector 60. The plurality of microservices 20 may be associated with a plurality of microservice caches 21. Preferably the database 10 is a centralised repository 11 which all

microservices are dependent on for data. The components of the self-correcting cache engine 4 may be managed and/or implemented by one or more programs run on one or more computing devices. The programs may be run on dedicated computers, processors, virtual machines, cloud computing platforms or any other device capable of software execution. One device may be used to run multiple different programs in parallel, and multiple devices may be used in combination to run the necessary programs. For example, the microservices 20, composite service 30 and error detector 60 may be implemented by programs run on dedicated virtual machines, while the message system may be implemented through a distributed message queue platform.

In preferred embodiments, the system 1 uses the microservices architecture as opposed to a monolithic or any other type of application architecture. Numerous variations of the microservices architecture are possible and the self-correcting cache engine 4 may be adapted for any such variant. In a microservices architecture, the system 1 comprises a plurality of microservices 20, wherein at least some of the microservices 20 have a data store which may be a dedicated database 10. One of the microservices may act as a centralised repository 11 having a database 10 that stores data required by other microservices 20. Other microservices 20 may then be dependent microservices that receive data from the centralised repository 11 and store it in their own data stores. Preferably the data stores of the dependent microservices 20 are microservice caches 21 that store processed data from the centralised repository 11. Each microservice 20 may be implemented by a program that manages the microservice cache 21 or any other data store, processes data, reads event messages or performs other tasks to implement the microservice 20. When one of the dependent microservices 20 fetches data from the centralised repository 11 it may process it into a data structure optimised for the purpose of the microservice 20. For example, a microservice 20 associated with a free text search may store data in an inverted index, while a microservice 20 associated with product information may store data in a key-value structure. Some data may be replicated between the microservice caches 21 of different microservices 20. Data stored in the microservice caches 21 may be assigned a Time To Live, after which the most recent data is fetched from the centralised repository 11. The microservices 20 may also fetch data upon prompting by a message received from the message system 50, wherein the event message may be received from a dedicated queue or other data structure of the message system specifically associated with the microservice 20.

In one embodiment, the centralised repository 11 is a type of microservice implemented by a program running on a processor and is connected to a database 10. The centralised repository 11 monitors changes to the database 10 and generates associated event messages. Preferably the database 10 is a relational database and is

stored using a persistent memory storage format such as hard disks or solid-state drives. The database 10 stores data associated with one or more external data providers such as vendors 2. Preferably the centralised repository 11 provides add, remove, search, and edit operations for modification of the data. Modification of the database 10 may be made by external data providers such as vendors 2. An Application Programming Interface (API) may be provided by the centralised repository 11 to facilitate such external modifications. In one embodiment, the centralised repository 11 may utilise APIs provided by the external data providers such as vendors 2 to make intermittent data requests, although it is more preferable that the APIs are provided by the centralised repository 11 to facilitate external modifications. The centralised repository 11 generates event messages for instances of data in the database 10 being modified. An event message may contain information including unique identifier of the product, timestamp, an original value of the data, a new value of the data, which microservices 20 use the data, or any other metadata associated with the data that was modified. For each instance of data modification, a single event message may be generated or multiple event messages may be generated. Where multiple event messages are generated, each may be associated with a different microservice 20. Preferably the event messages are immediately published to the message system 50 upon being generated. In a preferred embodiment, the external data providers are vendors 2 with products to be made available for sale on the online platform, and the data is inventory and pricing data for these products.

In one embodiment, the message system comprises a message data store for storing event messages and a program that manages event messages and ensures that they are read by the appropriate programs. Preferably the message data store is stored using a persistent storage format such as hard disks or solid-state drives. The message data store may be shared with the database 10 or it may be maintained on a separate drive. The message data store may be configured to store event messages in a data structure such as a queue. Preferably the message system data store is configured to store event messages in one or more First In First Out (FIFO) queues. Event messages may be published to the message data store from the centralised repository 11 or from another source that generates event messages. The message system 50 sends the event messages to one or more microservice caches and/or the composite service cache 31 in order to indicate that the associated data must be updated in the respective cache. The sending of event messages through the message system 50 may be implemented with a distributed message queue platform. In a preferred embodiment, the message data store is configured as a plurality of FIFO queues where each is associated with a microservice or composite service and the sending of event messages is achieved through a distributed message queue platform. A distributed message queue platform may be a platform such as Apache Kafka where messages are sent to nodes that store the messages in feeds that can

be subscribed to by listeners, configured to allow high throughput of messages in real time.

In one embodiment, the composite service cache 31 is managed by a program that processes data received from any of the centralised repository 11, event messages or other microservices 20. In embodiments where the online platform 1 uses the microservices architecture, the composite service cache program implements a composite service 30 that is dependent on a plurality of microservices 20. The composite service cache program reads in data from the centralised repository 11 or other services, processes it into data ready to be served to the user over the network, and stores it in the composite service cache 31. The composite service cache program may prefetch, process and store data in anticipation of it being needed to service a user request, thereby minimising response time. Alternatively, data may be fetched and stored in the composite service cache 31 in order to service a user request, reducing response time for subsequent requests for the same data. Processed data stored in the composite service cache 31 may be assigned a Time To Live, after which the most recent data is fetched from one of the centralised repository 11 or the appropriate service, which may be known as the "Pull Model". The composite service cache program may also fetch the most recent data from the centralised repository 11 or a microservice 20 upon prompting by an event message received from the message system 50, wherein the event message may be received from a dedicated queue or other data structure of the message system 50 specifically associated with the composite service cache 31. Processed data stored in the composite service cache 31 may take the form of one or more web pages to be displayed by a web browser, a data packet configured to be rendered for display by an installed application, or any other form of data packet configured to be easily displayed on the user device. In a preferred embodiment, the processed data stored in the composite service cache may be associated with the display of e-commerce user interfaces such as a home page, search page, product details page, account page, payment page or any other interface that might form part of an e-commerce user experience.

In one embodiment, the error detector 60 has a log data store 61 and is configured to analyse log data for cache error detection and correction purposes. The error detector 60 may implement its processes with dedicated programs in communication with each other or as subroutines of a single program. As seen in Figure 5, these programs or subroutines may include a similarity engine 62 which facilitates error detection, and/or a feedback engine 63 which facilitates error correction communications. The error detector 60 records activity of the user 3 in the log data store 61 as they view data on the online platform 1 through user interface 40. User activity logged may include actions performed by the user 3, pages visited by the user 3, data viewed by the user 3, and associated

timestamps or other metadata. The log data store 61 is preferably persistent memory storage such as hard disks or solid-state drives. The similarity engine 61 may continuously analyse the log data store 61 to match associated values displayed to the user 3 and identify any inconsistencies between them. In a preferred embodiment, the log data relates to e-commerce pages displayed to the user 3 such as a home page, search page, product details page, discovery page or payment page, and the values matched by the similarity engine 61 are values displayed on different pages that relate to the same attribute for a certain product. For example, the similarity engine 61 might identify that a certain product is displayed on both the search page and the discovery page. It may then also match which displayed values relate to this product's price, even if the data was displayed differently between the pages and/or was originally sourced from a different microservice 20. When the similarity engine 61 identifies inconsistencies between values such as stock count or price, the feedback engine 62 is notified and generates one or more correction messages that are sent to the message system 50. In generating the correction messages the feedback engine 62 may include information such as an identifier of the data that was inconsistent (for example a product ID) or information about which attributes were inconsistent (for example stock count or price). The correction messages may contain any or all of this information in addition to timestamp and/or other associated metadata. The correction messages may be sent to message queues or other data structures of the message system 50 associated with particular microservices 20 or the composite service 30.

The addition of the error detector 60 to a microservices architecture online platform 1 may serve to create higher accuracy of information delivered to the user 3 through the additional method of cache error correction it provides. Inconsistencies between the microservice caches 21 may arise through packet loss during network communication, processing errors in certain microservices 20, or differences in the timing of cache updates. An additional feedback path from the user interface 40 to the microservice caches 21 and especially the composite service cache 31 can eliminate inconsistencies between the caches before they would otherwise be corrected by slower means such as expiry of Time to Live windows or further updates to the database 10 causing event messages to be published. Thus, higher accuracy is achieved and hence improved user experience.

The present invention may be said to reside in a system for an online platform 1 having a self-correcting cache engine 4. The essential elements of the self-correcting cache engine 4 comprise a composite service 30 having a composite service cache 31, a message system 50 in communication with the composite service, one or more message sources in communication with the message system 50 and one or more data sources in

communication with the composite service 30. The composite service 30 fetches data from the one or more data sources and stores the data in the composite service cache 31, while the message system 50 receives messages from the one or more message sources and sends the messages to the composite service 30 to indicate that certain data in the composite service cache 31 must be updated by fetching it again from the one or more data sources. The message system 50 may thereby facilitate error correction of the composite service cache 31, which may furthermore be self-correcting based on the implementation of the message sources.

The cache may be self-correcting if the message sources form part of a feedback loop. The output of the self-correcting cache engine 4 may be to a user interface 40 served from the composite service cache 31, which forms part of the computing system for the online platform 1. Self-correction may be implemented where the message sources include an error detector 60 that is in communication with the user interface 40. The error detector 60 may log data displayed through the user interface 40 in a log data store 61 and identify inconsistencies within it, which may be sent as correction messages through the message system 50 to the composite service 30 thereby creating a feedback loop that corrects errors in the composite service cache 31. The one or more data sources may include a database 10 and/or a plurality of microservices 20, wherein the database 10 may be a centralised repository 11 upon which the plurality of microservices 20 are dependent to receive data. The database 10 may generate event messages whenever stored data is updated and be an additional message source of the message system 50, thereby providing another means through which data in composite service cache 31 is updated. The composite service 30 and/or the microservices 20 may be implemented by programs running on processors, which may be virtual machines. The microservices 20 may have microservice caches 31 which may be in communication with the message system and be prompted to update themselves by received messages. The message system 50 may be implemented using a distributed message queue platform.

In summary, the present invention may be said to reside in a computing system or system 1 that may comprise at least a self-correcting cache engine 4 and a user interface 40.

The self-correcting cache engine 4 may comprise a message system 50 that is configured to receive a stream of messages from at least one source 32, 33, store the messages and send the messages to at least one destination. The engine may also comprise a processor 35 (a composite service processor 35) implementing or running a composite service. The processor 35 may be in operative communication with at least one source 32, 33, and the message system 50. The composite service may be configured to fetch and process at least one data set from one or more sources 32, 33 and store the at

least one data set in a composite service cache that is part of or is operatively connected to the processor in a processed form. The composite service may also be able to receive the messages sent from the message system 50, use at least one content of the messages to fetch at least one updated data set from at least one of the sources 50, and use the at least one updated data set to update the at least one data set stored in the composite service cache 31. The composite service may also be configured to send the processed form of the at least one data set from the composite service cache.

The user interface may be configured to receive and display to a user the processed form of at least one data set sent from the composite service (composite service cache 31).

At least one source may be at least one of a message source that is the source of the messages, and a data source that is the source of the at least one data set.

The processed form of at least one data set from the composite service cache 31 may be sent over a network such as the internet. In one embodiment, the self-correcting cache engine 40 may further comprise at least one processor that is an error detection processor 65 which may be in operative communication with the user interface and the message system and implementing an error detector 60. The error detector 60 may be configured to detect inconsistencies in the processed form of at least one data set displayed by the user interface 40. The error detector 60 may also be configured to generate and send to the message system a correction message whenever an inconsistency is detected, wherein contents of the correction message is information about the at least one data set within which the inconsistency has been detected.

The error detector 60 may be configured to detect the inconsistencies through analysis of at least one log data set which may be collected during operation of the user interface and is stored in a log data store that is part of or is operatively connected to the error detection processor 65.

At least one data source 33 may include at least one processor that is a data source processor for implementing a plurality of microservices 20. The microservices may be configured to process and store some or all of the at least one data set fetched by the composite service 30.

One of the data source processors may implement a centralised repository 11 and be in operative communication with a database 10 configured to store some or all of the at least one data set fetched by the composite service 30.

A plurality of microservices 20 may be dependent microservices configured to be dependent on the centralised repository 11 for receiving and storing the at least one data set.

A plurality of microservices 20 may be configured to store data in a plurality of microservice caches 21.

The centralised repository 11 may be one message source 32 of the message system 50 and may be configured to generate an event message whenever the database 10 is updated. The contents of the event message (i.e. event message contents) may be information about the at least one updated data set.

At least one data set stored in the composite service cache 31 may be configured to expire after a set period of time, thereby triggering the composite service 30 to re-fetch and re-process at least one data set from at least one data source 33.

The centralised repository 11 may be configured to be updated by one or more external data providers.

The computing system is or may use or provide an e-commerce platform. One or more external data providers may be retail vendors 2. At least one data set may relate to inventory and pricing information for products available for sale.

The message system 50 may comprise one or more First In First Out (FIFO) queue structures which may each be associated with one of the plurality of microservices 20 or the composite service 30, and may be configured to store the messages until they are read at all destinations.

The composite service cache 31 and/or the microservices caches 21 may be implemented with a distributed Random-Access Memory (RAM) storage system.

The message system 50 may be implemented with a distributed message queue platform.

The present invention can be said to reside in a cache error correction system for a microservices architecture online platform 1. The system may comprise a processor 65 (error detection processor 65) implementing an error detector 60. The error detector may be configured to collect at least one log data set from a user interface of the online platform and store it in a log data store. The error detector 60 may continuously analyse at least one log data set stored in the log data store to identify one or more inconsistencies. The error detector 60 may generate and send a correction message whenever the one or

more inconsistencies are identified, wherein contents of the correction message is information about at least one data set that is inconsistent.

The error detector 60 may be further configured to detect the inconsistencies through analysis of at least one log data set which may be collected during operation of the user interface and is stored in a log data store that is part of or is operatively connected to the error detection processor 65.

Where in the foregoing description reference has been made to elements or integers having known equivalents, then such equivalents are included as if they were individually set forth.

Although the invention has been described by way of example and with reference to particular embodiments, it is to be understood that modifications and/or improvements may be made without departing from the scope or spirit of the invention.

CLAIMS

1. A computing system comprising:

a self-correcting cache engine comprising:

a message system configured to receive a stream of messages from at least one source, store the messages and send the messages to at least one destination;

a processor implementing or running a composite service, the processor being in operative communication with the at least one source, and the message system, the composite service being configured to:

fetch and process at least one data set relating to inventory and pricing information for products available for sale from the at least one source and store the at least one data set in a composite service cache that is part of or is operatively connected to the processor in a processed form;

receive the messages sent from the message system, use at least one content of the messages to fetch at least one updated data set from the at least one source, and use the at least one updated data set to update the at least one data set stored in the composite service cache; and

send the processed form of the at least one data set from the composite service cache to a user interface, wherein the user interface is part of the computing system and is configured to receive and display to a user the processed form of the at least one data set sent from the composite service;

wherein the self-correcting cache engine further comprises:

at least one error detection processor being in operative communication with the user interface and the message system, the at least one error detection processor implementing an error detector that is configured to:

detect inconsistencies in the processed form of the at least one data set displayed by the user interface; and

generate and send to the message system a correction message whenever an inconsistency is detected, wherein contents of the correction

message is information about the at least one data set within which the inconsistency has been detected,

wherein the error detector is configured to detect the inconsistencies through analysis of at least one log data set collected during operation of the user interface and stored in a log data store that is part of or is operatively connected to the at least one error detection processor, and wherein the at least one log data set comprises user activity data including actions performed by a user, pages visited by the user and data viewed by the user through the user interface.

2. The computing system as claimed in claim 1, wherein the at least one source is at least one of a message source, and a data source that is the source of the at least one data set.

3. The computing system as claimed in claim 2, wherein the one or more data sources include at least one processor that is a data source processor, the at least one data source processor implementing a plurality of microservices, wherein the microservices are configured to process and store some or all of the at least one data sets fetched by the composite service.

4. The computing system as claimed in claim 3, wherein one of the at least one data source processor implements a centralised repository and is operatively connected to a database configured to store some or all of the at least one data set fetched by the composite service.

5. The computing system as claimed in claim 4, wherein the plurality of microservices other than the centralised repository are dependent microservices configured to be dependent on the centralised repository for receiving and storing the at least one data set.

6. The computing system as claimed in any one of claims 3 to 5, wherein some of the plurality of microservices are configured to store data in a plurality of microservice caches.

7. The computing system as claimed in any one of claims 4 to 6, wherein the centralised repository is one of the at least one message sources of the message system and is configured to generate an event message whenever the database is updated, wherein contents of the event message are information about the at least one updated data set.

8. The computing system as claimed in any one of claims 2 to 7, wherein the at least one data set stored in the composite service cache is configured to expire after a set

period of time, thereby triggering the composite service to re-fetch and re-process the at least one data set from the one or more data sources.

9. The computing system as claimed in any one of claims 2 to 8, wherein the message system comprises one or more First In First Out (FIFO) queue structures.

10. The computing system as claimed in claim 9, wherein the one or more FIFO queue structures are each associated with one of the plurality of microservices or the composite service and are configured to store the messages until they are read at all destinations.

11. The computing system as claimed in any one of claims 2 to 10, wherein the composite service cache is implemented with a distributed Random Access Memory (RAM) storage system.

12. The computing system as claimed in any one of claims 2 to 11, wherein the message system is implemented with a distributed message queue platform.

13. A cache error correction system for a microservices architecture online platform that is an e-commerce platform, the system comprising:

a processor implementing an error detector,

the error detector configured to:

collect at least one log data set from a user interface of the online platform and store it in a log data store, wherein the at least one log data set comprises user activity data including actions performed by a user, pages visited by the user and products viewed by the user through the user interface;

continuously analyse the at least one log data set stored in the log data store to identify one or more inconsistencies; and

generate and send a correction message whenever the one or more inconsistencies are identified, wherein contents of the correction message is information about the at least one data set that is inconsistent.

14. The cache error correction system as claimed in claim 13, wherein the correction message is received by a message system of the online platform.

15. The cache error correction system as claimed in claim 13 or 14, wherein the message system sends the correction message to a secondary processor implementing a composite service, the composite service configured to:

store at least one data set received from one or more data sources in a composite service cache and receive the correction message from the message system,

use the contents of the correction message to fetch at least one updated data set from at least one of the one or more data sources, and

use the at least one updated data set to update the at least one data set stored in the composite service cache.

16. A method of cache error correction for an online platform that is an e-commerce platform comprising at least the following steps:

collecting at least one log data set as a user navigates the online platform and storing it in a log data store, wherein the at least one log data set comprises user activity data including actions performed by a user, pages visited by the user and products viewed by the user through the user interface;

continuously analysing the at least one log data set to match associated values and determine if the associated values are inconsistent;

identifying at least one data source that provides at least one data set associated with the inconsistency; and

sending one or more correction messages indicating at least one cache of the online platform to refresh the at least one data set corresponding to the inconsistency.

17. A method of cache error correction for an online platform that is an e-commerce platform comprising at least the following steps:

using a message system to receive a stream of messages from at least one source, store the messages and send the messages to at least one destination;

implementing or running a composite service in a processor that is in operative communication with the at least one source and the message system;

fetching and processing at least one data set from the at least one source and storing the at least one data set in a composite service cache that is part of or is operatively connected to the processor in a processed form, wherein the at least one data set relates to inventory and pricing information for products available for sale;

receiving the messages sent from the message system, using at least one content of the messages to fetch at least one updated data set from at least one source, and using

the at least one updated data set to update the at least one data set stored in the composite service cache;

sending the processed form of the at least one data set from the composite service cache;

using a user interface to receive and display to a user the processed form of the at least one data set sent from the composite service;

using at least one error detection processor in operative communication with the user interface and the message system to implement an error detector; and

using the error detector to:

detect inconsistencies in the processed form of the at least one data set displayed by the user interface; and

generate and send to the message system a correction message whenever an inconsistency is detected, wherein contents of the correction message is information about the at least one data set within which the inconsistency has been detected,

wherein the inconsistencies are detected through analysis of at least one log data set collected during operation of the user interface and stored in a log data store that is part of or is operatively connected to the at least one error detection processor, and wherein the at least one log data set comprises user activity data including actions performed by a user, pages visited by the user and data viewed by the user through the user interface.