(54) EVENT PROCESSING WITH ENHANCED THROUGHPUT

(57) The present systems and methods allow for rapid processing of large volumes of events. A producer node in a cluster determines a sharding key for a received event from an event stream. The producer node uses a sharding map to correlate the sharding key for the event with a producer channel, and provides the event to a producer event buffer associated with the producer channel. The producer event buffer transmits the event to a corresponding consumer event buffer associated with a consumer channel on a consumer node. The event processing leverages a paired relationship between producer channels on the producer node and consumer channels on the consumer node, so as to generate enhanced throughput. The event processing also supports dynamic rebalancing of the system in response to adding or removing producer or consumer nodes, or adding or removing producer or consumer channels to or from producer or consumer nodes.
CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Provisional Application No. 62/062,515 entitled “Internal Message Processing with Enhanced Throughput,” filed October 10, 2014, which is incorporated by reference herein in its entirety.

FIELD OF THE DISCLOSURE

[0002] The present disclosure relates to digital data processing, and more particularly, to methods, apparatus, and systems for processing events. The teachings herein have application, by way of non-limiting example, to rapid processing of events comprising large volumes of data between producer and consumer nodes in a cluster.

BACKGROUND

[0003] External systems generate growing numbers of events, such as social media posts and notifications. Furthermore, legacy systems can generate large volumes of events for processing by applications. Customers are increasingly looking to build or leverage enterprise applications that are capable of receiving and processing high volumes of events in short periods of time from these external or legacy systems. These systems can be used in a variety of applications ranging from health care to automotive repair. The applications can facilitate a range of business operations, from marketing and manufacturing to distribution to technical support. For example, an application can implement data-processing workflows to support processing transactional data ranging from customer service requests received by retail and banking enterprises, to routing and resolution of health care claims by insurance enterprises.

[0004] Computers can be networked into a cluster that includes two or more nodes capable of exchanging events or messages. The cluster can include one or more producer nodes and one or more consumer nodes. The producer nodes can transmit, or publish, events and the consumer nodes can receive, or subscribe to, events. Traditional techniques for exchanging and processing messages include a publish-subscribe model. In a publish-subscribe model, publishers or producers do not transmit messages direct to an individual subscriber or consumer. Instead, publishers transmit a message broadly to multiple subscribers at once. The subscribers generically register interest in certain messages and thereby receive only messages of interest.

[0005] However, the publish-subscribe model suffers from reduced throughput because a system uses valuable capacity in the network to provide a given message to multiple consumer nodes, in case more than one consumer node registers interest in the message. Furthermore, the publish-subscribe architecture generally uses an intermediate middleware node sometimes referred to as a message broker or message bus for message delivery, using additional resources in the cluster. A still further downside of traditional message processing systems is consumer nodes generally receive messages in any order, instead of preserving the order in which the producer nodes read the messages from a message stream for faster message routing and processing.

[0006] An object of this invention is to provide improved systems and methods for digital data processing. A more particular object is to provide improved systems and methods for event processing.

[0007] A further object is to provide such improved systems and methods as facilitate deployment of enterprise applications in a networked cluster of producer nodes and consumer nodes.

[0008] Yet a still further object is to provide such improved systems and methods as better utilize computing and networking resources upon addition or removal of producer nodes and consumer nodes to or from the cluster.

SUMMARY

[0009] The foregoing are among the objects attained by the invention which provides, in one aspect, a digital data processing system and method for event processing.

[0010] In some embodiments, the digital data processing system includes a producer node in communicative coupling with one or more consumer nodes and with a sharding map, for example over a network. The producer node can be configured to receive at least one event stream including a plurality of events, and determine a sharding key associated with an event among the plurality of events in the event stream. The producer node can be further configured to identify, based on the sharding map, a producer event buffer associated with a producer channel on the producer node for transmitting the event to a corresponding consumer event buffer associated with a consumer channel on a consumer node among the one or more consumer nodes. The sharding map can correlate the sharding key for the event with the producer channel. The producer node can be further configured to provide the event to the producer event buffer associated with the producer channel in order to transmit the event to the corresponding consumer event buffer associated with the consumer channel on the consumer node.
According to related aspects of the invention, the producer node can be further configured to initialize a plurality of producer channels. The producer node can reference a channel map that correlates an event in the event stream with one or more consumer channels on the one or more consumer nodes. The producer node can create at least one producer channel based on the channel map. The producer channel can be communicatively coupled with a corresponding consumer channel among the one or more consumer channels. The producer node can update the sharding map to correlate the sharding key for the event with the at least one producer channel created based on the channel map.

In related aspects of the invention, the sharding map can correlate the sharding key for the event with the producer channel based on a partition space. The partition space can be determined using a partition criterion based on a count of consumer nodes available to process the plurality of events in the event stream.

In further related aspects of the invention, the producer node can be configured to update the sharding map in response to detecting an update to the channel map. For example, the producer node can be configured to update the sharding map or the channel map, identify data to be moved to a different consumer channel or a different consumer node, and assign an overflow portion of the partition space based on the updated partition size to a new producer channel.

Alternatively, the redistribution of the partition space can be based on querying the one or more consumer nodes to identify resources available to the one or more consumer nodes, and weighting the partitions assigned to each producer node based on the identified resources.

Related aspects of the invention provide a producer node that can be further configured to adjust the rate of providing events to the producer event buffer, so as to manipulate the rate of events processed on the corresponding consumer node by the consumer event buffer, in order to improve throughput between the producer node and the consumer node.

In related aspects of the invention, the producer node can be further configured to bundle event transport metadata with the plurality of events in the event stream for transmission to the consumer node, and trigger the consumer node to execute a rule identified by the event transport metadata for processing the plurality of events. The update to the sharding map or the channel map can include adding a consumer node to the system, removing a consumer node from the system, adding one or more consumer channels to the consumer node, or removing one or more consumer channels from the consumer node.

In related aspects of the invention, the producer node can be further configured to transmit a message to the one or more consumer nodes in response to detecting the update to the sharding map or the channel map. The transmitted message can trigger the one or more consumer nodes to determine a delta that tracks the update to the sharding map or the channel map, identify data to be moved to a different consumer channel or a different consumer node, and copy the data to be moved to a cluster change data map. The data copied to the cluster change data map can trigger the consumer node to copy the moved data from the cluster change data map, clear the copied data from the cluster change data map, and update a status of the consumer node to active in a consumer map that tracks the one or more consumer nodes in the cluster. The producer node can be further configured to resume transmission of data to the consumer node in response to detecting the updated status of the consumer node in the consumer map.

The foregoing and other aspects of the invention are evident in the text that follows and in the drawings.

**BRIEF DESCRIPTION OF THE DRAWINGS**

**[0019]** Various objects, features, and advantages of the present disclosure can be more fully appreciated with reference to the following detailed description when considered in connection with the following drawings, in which like reference numerals identify like elements. The following drawings are for the purpose of illustration only and are not intended to be limiting of the invention, the scope of which is set forth in the claims that follow.

Figures 1A-1B illustrate example event processing systems, in accordance with certain embodiments of the present invention.

Figure 2A illustrates an example progression of an event through the event processing system, in accordance with certain embodiments of the present invention.

Figure 2B illustrates an example method for event processing, in accordance with certain embodiments of the present invention.

Figures 3A-3B illustrate an example channel map and sharding map for event processing, in accordance with certain embodiments of the present invention.

Figures 4A-4B illustrate an example consumer node for event processing, in accordance with certain embodiments of the present invention.

Figures 5A-5B illustrate an example producer node for event processing, in accordance with certain embodiments of the present invention.
The present systems and methods allow for rapid processing of large volumes of discretely generated data referred to herein as events. In some embodiments, the event processing systems and methods can be general enough to be used as a generic message passing service in a cluster of one or more producer nodes and consumer nodes. (The event processing system and related methods, as discussed here and elsewhere herein, are referred to elsewhere herein simply as "the event processing.") The event processing can exhibit and support both horizontal as well as vertical scaling to achieve speed and robustness in current and future client installations.

The event processing systems and methods involve receiving an event stream having one or more events. A producer node in a cluster determines a sharding key for a received event (i.e., a key for grouping events for routing). The producer node uses a sharding map to correlate the sharding key for the event with a producer channel. The producer node provides the event to a producer event buffer associated with the producer channel. The producer event buffer transmits the received event directly to a corresponding consumer event buffer associated with a consumer channel on a consumer node. The event processing leverages a paired relationship established between one or more producer channels on the producer node, and one or more consumer channels on the consumer node to generate enhanced throughput compared with traditional messaging systems.

The event processing systems and methods have utility in, for example, rapidly processing discussions on social media. Today, many people use and communicate ideas through social media. The topics and ideas shared can vary greatly, however topics specific to an industry or business are potentially germane to an event processing system of the type referred to above (and described further herein) employed by a given enterprise, say, in that industry or business. As users of social media publish discussions, these discussions can form the basis for grass roots marketing and decision-based campaigns, e.g., by that enterprise. In some embodiments, such event processing allows the enterprise to tap into these theme-based discussions to "mine" information important to their industry or specific business. The event processing provides an ability to tap into this vast source of data and distill the data into a meaningful form. To achieve this goal, some embodiments of the event processing system include a producer node that is able to connect to a "fire-hose" of events, or data, streaming from event services. Non-limiting example event services include microblogs such as Twitter, personal social networks such as Facebook, and professional social networks such as LinkedIn. In some embodiments, the event processing systems and methods provide a mechanism to identify events declaratively in which the enterprise is interested, once an administrator configures and establishes a connection. Further embodiments of the event processing allow the enterprise to define declarative rules that describe how to process those events of interest.

The event processing systems and methods described herein also have utility in rapidly processing events from traditional data warehouses or so-called "Big Data" systems that generate large volumes of data in short periods of time. A non-limiting example includes processing insurance claim data using declarative rules. In this case a large volume of insurance claim data can flow into the event processing system. The event processing routes the events from event services to producer nodes and consumer nodes for processing, transformation, and potentially storage.

System Architecture
and consumer nodes in communication over any number of networks within the scope of the invention.

Non-limiting examples of event services 102a-c include microblog 102a, personal social network 102b, and professional social network 102c, all of the type known in the art as adapted in accord with the teachings hereof. In this regard, event services 102a-c can be implemented, for example, on one or more digital data processing systems in the conventional manner known in the art, again, as adapted in accord with the teachings hereof. Event services 102a-c provide an event stream to producer nodes 104a-b. In some embodiments, the event stream can include events aggregated from multiple event sources.

Producer nodes 104a-b communicate with event services 102a-c over network 108. Producer node 104a includes event processor 110a, and producer node 104b includes event processor 110b. Event processors 110a-b are configured to receive events from one or more event services 102a-c, process, and route those events to one or more of consumer nodes 106a-c. In some embodiments, producer nodes 104a-b can participate in and communicate with the overall cluster of illustrated nodes in support of client interactions, while the producer nodes themselves remain isolated and hidden from end users who are using consumer nodes 106a-c. That is, event processing system 100 allows administrators to configure a system in which the end users are only aware that system 100 receives events from event services 102a-c and routes the received events to consumer nodes 106a-c, without requiring knowledge of architectural or implementation details of-or even the existence of-producer nodes 104a-b.

Consumer nodes 106a-c receive the routed events from producer nodes 102a-b over network 108. Consumer node 106a includes event processor 112a, and consumer node 106b includes event processor 112b. Event processors 112a-b are configured to receive events from producer nodes 104a-b and process the events. Consumer nodes 106a-c participate and communicate with the overall cluster of illustrated nodes supporting client interactions and can be allocated for connection to and communication with end users. In some embodiments, consumer nodes 106a-c process received events in accordance with event transport metadata that is bundled with the event. For example, the event transport metadata can trigger consumer nodes 106a-c to process received events by performing sentiment analysis on the received events, grouping or aggregating the received events for further processing, displaying user interface updates about the received events, or storing the received events in a database.

In some embodiments, producer nodes 104a-b can include one or more producer digital data processors, and consumer nodes 106a-c can include one or more consumer digital data processors. The producer and consumer digital data processors can be digital processors of the type commercially available in the marketplace suitable for operation in event processing system 100 and adapted in accord with the teachings hereof, for example, utilizing rules forming applications executing in one or more rules engines, e.g., as discussed elsewhere herein. Though producer nodes 104a-b and consumer nodes 106a-c can be typically implemented in server-class computers such as a minicomputer, producer nodes 104a-b and consumer nodes 106a-c may also be implemented in desktop computers, workstations, laptop computers, tablet computers, personal digital assistants (PDAs) or other suitable apparatus adapted based on the systems and methods described herein. The producer digital data processor and consumer digital data processor include central processing, memory, storage using a non-transitory computer-readable medium (e.g., a magnetic disk, solid state drive, or other storage medium), and input/output units and other constituent components (not shown) of the type conventional in the art that are programmed or otherwise configured in accord with the teachings hereof.

Network 108 can include one or more networks of the type commercially available in the marketplace or otherwise suitable for supporting communication between event services 102a-c, producer nodes 104a-b, and consumer nodes 106a-c in accord with the teachings hereof. Network 108 can be wired or wireless, a cellular network, a Local Area Network (LAN), a Wireless LAN (WLAN), a Metropolitan Area Network (MAN), a Wireless MAN (WMAN), a Wide Area Network (WAN), a Wireless WAN (WWAN), a Personal Area Network (PAN), a Wireless PAN (WPAN), or a network operating in accordance with existing IEEE 802.11, 802.11a, 802.11b, 802.11g, 802.11n, 802.16, 802.16d, 802.16e, 802.16m standards or future versions or derivatives of the above standards.

Figure 1B illustrates example event processing system 114 in accordance with certain embodiments of the present invention. Event processing system 114 illustrates event services 102a-c in communication with producer nodes 104a-b and consumer nodes 106a-c over network 134. In some embodiments, producer nodes 104a-b use channel map 116 and sharding map 118 to transmit events from one or more of event services 102a-c to one or more of consumer nodes 106a-c over network 134. In further embodiments, producer nodes 104a-b and consumer nodes 106a-c can be in communication with rules engine 128 over network 134.

An event service 102a-c publishes an incoming event into the cluster, and event processing system 114 ultimately designates a consumer node 106a-c in the cluster to handle or process the event (e.g., using one of producer nodes 104a-b). Some embodiments of event processing system 114 can support additional features. For example, producer nodes 104a-b and consumer nodes 106a-c can process events in the order in which event services 102a-c produce the events. An event key associated with events can identify that an event belongs to a particular group of related events. Lastly, event processing system 114 can allow an administrator to configure the system so that the same consumer node 106a-c processes events that have the same event key (e.g., via one or more of event processors 110a-b on producer nodes 104a-b).
In some embodiments, event processors 110a-b on producer nodes 104a-b process events using embedded Hazelcast logic (hereinafter, simply, "Hazelcast") 124a-b, and event processors 112a-c process events using embedded Hazelcast 126a-c. As those skilled in the art will appreciate, Hazelcast refers to a publicly available third-party architecture and processing mechanism for in-memory object caching. Event processing system 114 can leverage Hazelcast to enable distributed object caching and locking, and node-to-node messaging. A producer node or consumer node using Hazelcast can become what the Hazelcast architecture refers to as a Hazelcast node. Hazelcast allows automatic discovery of other Hazelcast nodes using discovery mechanisms such as multicast or direct Internet Protocol (IP) discovery.

Producers 104a-b use interfaces 120a-b to receive events from event services 102a-c over network 134. In some embodiments, interfaces 120a-b use embedded Hazelcast 124a-b to connect producer nodes 104a-b with event services 102a-c and consumer nodes 106a-c, for example using a discovery mechanism based on TCP/IP.

In some embodiments, producer nodes 104a-b and consumer nodes 106a-c process events using channel map 116 and sharding map 118. Channel map 116 correlates an event in the event stream from event services 102a-c with consumer nodes 106a-c. Specifically, channel map 116 identifies consumer channels on consumer nodes 106a-c to which to route and process events. Sharding map 118 correlates a sharding key for an event (e.g., a key for grouping or clustering events for routing) with a producer channel on producer nodes 104a-b to which to route and process events. As a consequence of the foregoing and the actions below, event processing system 114 creates a paired relationship between producer channels on producer nodes 104a-b and consumer channels on consumer nodes 106a-c to enable rapid processing of large volumes of events from event services 102a-c. In some embodiments, channel map 116 and sharding map 118 may be distributed in the cluster. For example, further embodiments of channel map 116 and sharding map 118 may be implemented in event processing system 114 as Hazelcast distributed maps.

In some embodiments, producer node 104a uses embedded Hazelcast 124a to read and write entries from channel map 116 and sharding map 118 over interface 120a, and producer node 104b uses embedded Hazelcast 124b to read and write from channel map 116 and sharding map 118 over interface 120b. Similarly, consumer nodes 106a-c use embedded Hazelcast 126a-c to read and write from channel map 116 and sharding map 118 over interfaces 122a-c.

In some embodiments, an enterprise can deploy event processing system 114 in support of enterprise applications executing locally on or remote to the cluster of illustrated nodes. Such enterprise applications can include specialized software or hardware used within a specific industry or business function (e.g., human resources, finance, healthcare, telecommunications, insurance, etc.). Alternatively, the enterprise applications can include cross-industry applications (e.g., project management), or other types of software or hardware applications.

In some embodiments, rules may define the enterprise applications. Producer nodes 104a-b and consumer nodes 106a-c can be in communication with rules engine 128. Rules engine 128 can be communication with rules base 130 and transactional database 132. As the application executes on a producer digital data processor (e.g., producer nodes 104a-b) or a consumer digital data processor (e.g., consumer nodes 106a-c), event processing system 114 may retrieve any portion of the rules that define the application from rules base 130 and process or execute the rules in response to requests or events signaled to or detected by the producer digital data processors or consumer digital data processors at run-time, (e.g., using rules engine 128).

Rules base 130 can include a rules base of the type known in the art (albeit configured in accord with the teachings hereof) for storing rules (e.g., scripts, logic, controls, instructions, metadata, etc.) and other application-related information in tables, database records, database objects, and so forth. Preferred rules and rules bases can be of the type described in United States Patent Nos. 5,826,250, entitled “Rules Bases and Methods of Access Thereof” and 7,640,222, entitled “Rules Base Systems and Methods with Circumstance Translation,” the entire contents of both of which are incorporated by reference herein in their entirety. In other embodiments, rules and rules bases that are architectured or operated differently may be used as well.

Some embodiments may utilize multiple rules bases. For example, rules base 130 may be an enterprise-wide rules base in communication with rules engine 128, and domain-specific rules bases may be accessible to producer nodes 104a-b or consumer nodes 106a-c via network 134. If multiple rules bases are provided in a given embodiment, the rules bases may be of like architecture and operation or may differ in architecture and operation as well.

In some embodiments, rules may comprise meta-information structures. For example, the rules can include data elements or method elements. The method elements can be procedural or declarative. For example, method elements in a rule may be procedural insofar as the rule comprises one or more of a series of ordered steps. Declarative elements in a rule may set forth (i.e., declare) a relation between variables or values (e.g., a loan rate calculation or a decision-making criterion). Alternatively, declarative elements may declare a desired computation or result without specifying how the computations should be performed or how the result should be achieved. In one non-limiting example, a declarative portion of a rule may declare a desired result of retrieving a specified value without specifying a data source for the value or a particular query language for such retrieval (e.g., SQL, QCL, QL, etc.). In other cases, the declarative portion of a meta-information structure may comprise declarative programming language statements (e.g., SQL). Still other types of declarative meta-information structures are possible.
While some rules may comprise meta-information structures that are wholly procedural and other rules may comprise meta-information structures that are wholly declarative, event processing system 114 can also include rules that comprise both procedural and declarative meta-information structures. That is, such rules can have meta-information structure portions that are declarative, as well as meta-information structure portions that are procedural. Furthermore, rules of the illustrated embodiments that comprise meta-information structures may also reference or incorporate other rules. Those other rules may themselves in turn reference or incorporate still other rules. As a result, editing such a rule may affect one or more rules that incorporate it (if any).

An advantage of rules that comprise meta-information structures over conventional rules is that meta-information structures provide administrators with flexibility to apply code-based or model-driven techniques in development and modification of applications or computing platforms. Particularly, like models in a model-driven environment, meta-information structures comprise data elements that can be used to define aspects of a complex system at a higher level of abstraction than source code written in programming languages such as Java or C++. On the other hand, administrators may also embed programming language statements into meta-information structures if the administrators deem that to be the most efficient design for the system being developed or modified. At run-time, rules engine 128 can convert the data elements of the meta-information structures along with programming language statements (if any) automatically into executable code for the application.

Thus, in some embodiments rules may be the primary artifacts that get created, stored (e.g., in rules base 130) or otherwise manipulated to define or modify the overall functionality of rules-based applications. The applications may automate or manage various types of work in different business domains at run-time. By way of non-limiting example, rules stored in rules base 130 may be configured to define aspects of an application. For example, rules can define the user interface, decision logic, integration framework, process definition, data model, reports, or security settings of a given application.

Transactional database 132 can include databases of the type known in the art (albeit configured in accord with the teachings hereof) for storing corporate, personal, governmental, or other data. Rules such as in rules base 130 may generate, update, transform, delete, store, or retrieve the data (herein collectively referred to as “processing” the data). Example data may include financial data; customer records; personal data; design-time, development-time, or runtime data related to an application; or other types of data. Transactional database 132 may store the data in tables, database records, or database objects, for example.

Transactional database 132 may be present in any given embodiment. Conversely, some embodiments may use multiple transactional databases, e.g., an enterprise-wide database on producer nodes 104a-b and branch-office specific databases on consumer nodes 106a-c, by way of non-limiting example. If multiple transactional databases are provided in a given embodiment, the transactional databases may be of like architecture and operation; though, they may have differing architecture or operation, as well.

Rules engine 128 can be of the type conventionally known in the art (albeit configured in accord with the teachings hereof) for use in processing or executing rules from rules base 130 to process data in (or for storage to) transactional database 132, e.g. in connection with events signaled to or detected by rules engine 128. Preferred such rules engines are of the type described in United States Patent No. 5,826,250, entitled “Rules Bases and Methods of Access Thereof,” United States Patent No. 7,640,222, entitled “Rules Base Systems and Methods with Circumstance Translation,” and United States Patent No. 8,250,525, entitled “Proactive Performance Management For Multi-User Enterprise Software Systems,” all of which are incorporated by reference in their entirety herein. Rules engine 128 may be implemented in a single software program, multiple software programs or modules, or a combination of software modules or programs. Rules engine 128 may comprise programming instructions, scripts, or rules (e.g., rules stored in rules base 130) or a combination thereof.

Some embodiments of rules engine 128 may execute on or over multiple digital data processors. For example, event processing system 114 may invoke rules engine 128 for execution on a single digital data processor (e.g., a digital data processor on a producer node 104a-b or a consumer node 106a-c). Subsequently, event processing system 114 may apportion, distribute, or execute portions of rules engine 128 (or, potentially, the entirety of rules engine 128) over multiple digital data processors.

Other ways of implementing or executing rules engine 128 are also possible. By way of non-limiting example, rules engine 128 may have additional distinct components or portions that can be apportioned and distributed separately. Non-limiting example components can include a data access component for processing data during rule execution, a session management component for keeping track of activity across sessions of interaction with a digital data processor, or a performance monitoring component for monitoring and interacting with various system resources or event logs to manage performance thresholds.

Finally, network 134 can include one or more networks for supporting communication between event services 102a-c, producer nodes 104a-b, consumer nodes 106a-c, and rules engine 128. Network 134 can be wired or wireless, a cellular network, a Local Area Network (LAN), a Wireless LAN (WLAN), a Metropolitan Area Network (MAN), a Wireless MAN (WMAN), a Wide Area Network (WAN), a Wireless WAN (WWAN), a Personal Area Network (PAN), a Wireless...
Figure 2A illustrates an example progression of an event through the event processing system, in accordance with certain embodiments of the present invention. The example progression illustrates metadata added to an event using the event processing methods from event services 102a-c to producer nodes 104a-b and ultimately to consumer nodes 106a-c.

The progression begins with event services 102a-c serving an event stream. For example, the event stream may be a series of social media microblog posts, or tweets, from microblog service Twitter. Each individual social media post can be considered an event to the event processing system. In some embodiments, an event can contain data and metadata. Non-limiting example data includes the contents of the event. An example tweet may include contents such as "I got great service at this restaurant today!" Events may also be collected into groups of related events. For example, a series of social media posts can describe a progression of events at a party. Events can also include metadata. As used herein, the term metadata refers to attributes that the event processing considers and processes separately from the underlying data in the event. Non-limiting example metadata includes the author of the social media post, a date of the social media post, an event stream type, and an event stream name.

In some embodiments, the event stream type can describe a source of the event stream. Non-limiting example event stream types can include social media services, such as Twitter, Facebook, or LinkedIn. In some embodiments, the event stream name can describe a specific instance of a stream type. The event processing supports processing multiple event streams in the cluster. The event stream name allows the event processing to address named event stream instances separately. Non-limiting example event stream names can include Finance Department, Human Resources Department, Department A, or Department B.

Producer nodes 104a-b receive the event stream from event sources 102a-c. Producer nodes 104a-b proceed to shard, or partition, the event stream into individual events for routing to consumer nodes 106a-c using the mechanisms discussed in detail below. In addition to the existing data and metadata associated with the events (e.g., contents, author, stream type, stream name), producer nodes 104a-b can also generate metadata. For example, producer nodes 104a-b can generate bundle event transport metadata with received events. The event transport metadata can identify rules for the consumer node to execute upon receiving the sharded event, for example in conjunction with a rules engine. Using these identified rules, the event transport metadata can instruct the consumer node how to process the received event.

In some embodiments, in processing and routing the received event, the producer node determines various identifiers (e.g., using an event processor). For example, the producer node determines a producer channel for routing the event to a given consumer node, using a sharding key associated with the event. The sharding key groups or clusters events for routing from the producer node to the consumer node. In some embodiments, the producer node determines the sharding key based on the contents of the event. The event processing system creates a paired relationship between the producer channel and a corresponding consumer channel on a consumer node. In some embodiments, the producer node can modulate the rate of providing events to the producer channel, as a manner of flow control to improve throughput to the consumer node.

Consumer nodes 106a-c receive and process the transmitted event from producer nodes 104a-b. The data and metadata associated with the received event remain the same between consumer nodes 106a-c and producer nodes 104a-b. In some embodiments, consumer nodes 106a-c retrieve the bundled event transport metadata from the event. The event transport metadata identifies for consumer nodes 106a-c how to process the received event. In further embodiments, the event transport metadata can identify rules that are accessible to consumer nodes 106a-c. Non-limiting example rules can include rules to perform sentiment analysis on the received events, group or aggregate the received events for further processing, display user interface elements associated with or about the received events, or store the received events in a database. Consumer nodes 106a-c then execute the rules to process the received event, potentially in conjunction with a rules engine.

Figure 2B illustrates example method 290 for event processing, in accordance with certain embodiments of the present invention. A producer node receives an event stream from one or more event services (step 292). In some embodiments, the event stream can include events aggregated from multiple event services. The producer node shards, or partitions, the event stream into individual events, e.g., as described elsewhere herein.

The producer node then determines a sharding key for each individual event (step 294). In some embodiments, the producer node determines the sharding key as follows. An administrator initially configures a partition space for a given event stream. As used herein, the term partition space refers to a universe or number of chunks or partitions into which a given event stream will be divided. For example, an event stream of stream type Twitter and stream name Department A might have a partition space of 100. Another event stream of stream type Facebook and stream name Department B might have a partition space of 200. Each event in the event stream can have an associated event key. In some embodiments, the producer node can determine the event key based on the contents of the event. For example, the event key can be a hash code or fingerprint of the contents of the event. The producer node can determine the
sharding key by computing a modulo of the event key for an individual event with the partition space for a given event stream. As a result, the sharding key will be a value between 0 and the partition space minus one. For example, for a partition space of 100, the computation of the sharding key results in a value from 0-99. For a partition space of 200, the computation results in a value from 0-199.

[0058] The producer node identifies a producer event buffer associated with a producer channel using the sharding key and sharding map (step 296). The sharding map correlates a sharding key with a producer channel. In some embodiments, prior to receiving an event the producer node initializes and creates these producer channels using a list of consumer channels provided by the channel map. The producer node initializes a producer event buffer for each producer channel. The producer node also creates a paired relationship between the producer channel on the producer node and a corresponding consumer channel on a consumer node. The consumer channel also has its own associated consumer event buffer.

[0059] The producer node then provides the event to the producer event buffer associated with the producer channel identified by the sharding map (step 298). The producer node has previously configured a paired relationship between the producer channel identified by the sharding map and a corresponding consumer channel on a desired consumer node. Accordingly, providing the event to the producer event buffer allows the producer node to transmit the received event to the corresponding consumer event buffer associated with the consumer channel on the consumer node. In some embodiments, the producer node can modulate the rate of providing events to the producer channel, so as to improve throughput between the producer node and consumer node using dynamic flow control.

[0060] Figure 3A illustrates example channel map 116 for event processing, in accordance with certain embodiments of the present invention. Channel map 116 tracks an inventory or universe of consumer channels available across all consumer nodes in the cluster for event processing. Channel map 116 illustrates seven example consumer channels, consumer channels 0-6 (302a-c, 304a-d). Although Figure 3A includes seven consumer channels for illustrative purposes, the event processing can be used with any number of consumer channels across any number of consumer nodes.

[0061] Some embodiments of the event processing system associate each event stream from the event services with a given number of channels. Channels represent resources available on producer nodes and consumer nodes to receive, transmit, and process events. For example, a first event stream for stream type Twitter and stream name "Department A" may have three channels available to process the event stream. In some embodiments the event processing system implements the three channels as three producer channels on one or more producer nodes, in a paired relationship with three consumer channels on one or more consumer nodes. Similarly, a second event stream for stream type Twitter and stream name "Department B" may have four channels available which the event processing system may implement as four producer channels on one or more producer nodes, in a paired relationship with four consumer channels on one or more consumer nodes. Channel map 116 illustrates this example arrangement of three and four consumer channels 302a-c, 304a-d.

[0062] In some embodiments, channel map 116 tracks the following information per consumer channel:

- **Stream type**: A stream type represents the general type of the event stream. A non-limiting example stream type is "Twitter." In some embodiments, the event processing system uses the stream type in the description of the event format which describes the properties of the event for a given stream type.
- **Stream name**: A stream name is a named instance of a given stream type. A non-limiting example stream name is "Finance department," "Human resources department," "Department A," or "Department B." Some embodiments of the event processing system allow receiving and processing multiple event streams in the cluster. To keep the multiple event streams logically separate during event processing, the event processing system supports named instances that can be addressed based on stream name.
- **NodeID**: Node ID represents the consumer node on which a particular consumer channel resides. In some embodiments, the node ID is a Hazelcast universally unique identifier (UUID) for the consumer node on which a Hazelcast topic for this consumer channel resides (see channel name below).
- **Channel name**: Channel name allows a name to be associated with a consumer channel instance. In some embodiments, the channel name is implemented as a Hazelcast topic name.
- **Status**: The status tracks a current status for the consumer channel. For example, the status can indicate whether or not a given consumer channel on a given consumer node is currently active, or offline for maintenance.

[0063] Channel map 116 allows the event processing to process multiple event services, such as multiple Twitter streams for different named instances of event services (e.g., different stream type and stream name). Channel map 116 tracks a first set of entries for an event stream having stream type Twitter and stream name Department A, and a second set of entries for stream type Twitter and stream name Department B. Channel map 116 indicates that three consumer channels are available to process the first event stream with stream type Twitter and stream name Department A. Consumer channels 0-2 (302a-c) show a status of Active. Furthermore, consumer channels 0-2 (302a-c) are allocated across two different consumer nodes. The first consumer node has node ID U1234567 and has two consumer channels.
A. The second consumer node has node ID U55555555 and has one consumer channel 2 (302c) active for processing events from the first event stream, for stream type Twitter and stream name Department A. The remaining two consumer channels 3-4 (304a-b) are active for processing events from the second event stream, for stream type Twitter and stream name Department B. Similarly, channel map 116 indicates that the second consumer node with node ID U55555555 has three consumer channels in total. The first consumer channel 2 (302c) is active for processing events from the first event stream, for stream type Twitter and stream name Department A. The remaining two consumer channels 5-6 (304c-d) are active for processing events from the second event stream, for stream type Twitter and stream name Department B.

The event processing uses channel map 116 to discover dynamically what event streams are active and supported in the cluster. Furthermore, channel map 116 allows the event processing to query the channel map for an event stream of interest (e.g., by stream type and stream name), and discover a list of consumer channels and consumer nodes that are available to process events from that event stream of interest (e.g., have an Active status).

Some embodiments of channel map 116 also support elasticity in the cluster. The elasticity can include horizontal and vertical scaling. Horizontal scaling refers to support by the event processing for dynamically adding or removing producer and consumer nodes to or from the cluster. Vertical scaling refers to support for dynamically adding or removing consumer channels to or from an individual consumer node, or producer channels to or from an individual producer node. In some embodiments, when a consumer node first comes online and joins the cluster (i.e., horizontal scaling), the consumer node registers its supported event streams and consumer channels with channel map 116. In further embodiments, the producer nodes and consumer nodes can register event listeners and item listeners with channel map 116. In response to a notification of a change to channel map 116, the producer nodes or consumer nodes can react to the changes. Example changes to channel map 116 include adding new channels, removing channels, or updating node IDs, channel names, status, or other fields within a channel. Example reactions to these changes can include a producer node creating corresponding producer channels in response to receiving notifications of new consumer channel entries in channel map 116 from a consumer node. For example, a new consumer node may join the cluster (e.g., horizontal scaling), or an existing consumer node may add support for additional consumer channels (e.g., vertical scaling). In either case, the producer and consumer nodes in the cluster can react accordingly to changes to channel map 116 in response to update notifications from the event listeners and item listeners.

Some embodiments of channel map 116 also support elasticity in the cluster. The elasticity can include horizontal and vertical scaling. Horizontal scaling refers to support by the event processing for dynamically adding or removing producer and consumer nodes to or from the cluster. Vertical scaling refers to support for dynamically adding or removing consumer channels to or from an individual consumer node, or producer channels to or from an individual producer node. In some embodiments, when a consumer node first comes online and joins the cluster (i.e., horizontal scaling), the consumer node registers its supported event streams and consumer channels with channel map 116. In further embodiments, the producer nodes and consumer nodes can register event listeners and item listeners with channel map 116. In response to a notification of a change to channel map 116, the producer nodes or consumer nodes can react to the changes. Example changes to channel map 116 include adding new channels, removing channels, or updating node IDs, channel names, status, or other fields within a channel. Example reactions to these changes can include a producer node creating corresponding producer channels in response to receiving notifications of new consumer channel entries in channel map 116 from a consumer node. For example, a new consumer node may join the cluster (e.g., horizontal scaling), or an existing consumer node may add support for additional consumer channels (e.g., vertical scaling). In either case, the producer and consumer nodes in the cluster can react accordingly to changes to channel map 116 in response to update notifications from the event listeners and item listeners.

Figure 3B illustrates example sharding map 118 for event processing, in accordance with certain embodiments of the present invention. Some embodiments of sharding map 118 correlate a partition for a given event from the event stream with a producer channel on the producer node. The producer node is then able to provide the event into the producer channel which transmits the event to a corresponding consumer channel on the consumer node. Sharding map 118 includes channel-partitions map 306 and partition-to-channel index 310.

Sharding map 118 correlates partitions for a given event with a producer channel on the producer node. In some embodiments, an administrator configures a partition space for each event stream of interest. The partition space defines a count of discrete chunks that events will be broken into, for a given event stream. In some embodiments, the event system can determine the partition space according to a partition criterion. For example, the producer node can identify based on the channel map a count of consumer nodes available to process the event stream. Upon detecting a relatively higher count of consumer nodes available, the event system can suggest increasing the partition space, to reflect increased capacity for processing the events in the event stream. Upon detecting a relatively lower count of consumer nodes, the event system can suggest decreasing the partition space to reflect decreased capacity for event processing. Alternatively, in some embodiments the partition space can have a predefined default value of 100. Based on the default value of 100, the producer node can partition received events into a value between 0-99 using sharding map 118. For example, if the partition space has a value of 50, the producer node can assign received events a sharding key between 0-49. If the partition space has a value of 200, the producer node can assign a sharding key between 0-199.

When the producer node receives an event stream from one or more event services, the producer node uses sharding map 118 to determine a sharding key for individual events in the event stream and route the events to a producer channel for transmission. In some embodiments, the producer node determines the sharding key as follows. The producer node determines an event key for an individual event. For example, the producer node may determine the event key based on the contents of the event, such as determining a hash code (or fingerprint) of the event. The producer node determines the appropriate partition by calculating a modulo of the event key with the partition space. The producer node looks up the resulting partition in partition-to-channel index 310. For example, if the resulting partition has a value of three, partition-to-channel index 312a would apply and the producer node would provide the event to producer channel 0 (314a). If, instead, the resulting partition has a value of 52, partition-to-channel index 312b would apply and the producer node would provide the event to producer channel 1 (314b). If the resulting partition has a value of 49, partition-to-channel index 312c would apply and the producer node would provide the event to producer channel 2 (314c).
In some embodiments, the event processing system creates and populates sharding map 118 based on channel map 116 according to a sharding algorithm executing, e.g., on one or more of the nodes. With reference to Figure 3A, during initialization the producer node establishes a paired relationship between producer channels 0-2 (314a-c) and consumer channels 0-2 (302a-c) tracked in channel map 316. That is, when a producer node comes online and joins the cluster, the producer node uses the sharding algorithm to update sharding map 118 based on channel map 116 and the partition space, creates producer channels 0-2 (314a-c) and consumer channels 0-2 (302a-c) tracked in channel map 316.

In one embodiment, the sharding algorithm evenly distributes the partition space uniformly across available producer channels. For example, if the partition space is 100 and there is one producer channel active, the partition-to-channel index would reflect:

<table>
<thead>
<tr>
<th>Producer Channel</th>
<th>Partitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>producer channel 0</td>
<td>partitions 0-99</td>
</tr>
</tbody>
</table>

In response to a notification that a consumer node has added another active consumer channel, the producer node can expand dynamically and create another producer channel (i.e., vertical scaling). In this embodiment of the sharding algorithm, the partition-to-channel index would become:

<table>
<thead>
<tr>
<th>Producer Channel</th>
<th>Partitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>producer channel 0</td>
<td>partitions 0-49</td>
</tr>
<tr>
<td>producer channel 1</td>
<td>partitions 50-99</td>
</tr>
</tbody>
</table>

The producer node can also populate the channel-partitions map and the partition-to-channel index using these results from the sharding algorithm.

Another embodiment of the sharding algorithm populates sharding map 118 by distributing the partition space dynamically using an overflow portion in response to updates to the available consumer channels. In this embodiment, the sharding algorithm determines an updated partition size based on an updated count of available consumer channels, and assigns overflow portions of the partition space and the updated partition size to a newly added producer channel.

For example, when a consumer node comes online and joins the cluster, the consumer node may advertise one active consumer channel (channel 0) for an event stream of interest. When the producer node receives a notification of the newly active consumer channel, the producer node updates sharding map 118 by assigning all partition values of the partition space to a corresponding new producer channel:

<table>
<thead>
<tr>
<th>Producer Channel</th>
<th>Partitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>producer channel 0 (new)</td>
<td>partitions 0-99</td>
</tr>
</tbody>
</table>

Subsequently, the consumer node may elect to expand the number of consumer channels available, for example by adding a second consumer channel. In response to a notification of the newly active consumer channel, the producer node may redistribute the partition space. The producer node determines an updated partition size based on an updated count of available consumer channels. In some embodiments, the producer node calculates

\[ pz = \frac{ps}{c} \]

Equation (1)

where \( pz \) refers to the updated partition size, \( ps \) refers to the size of the partition space, and \( c \) refers to the updated number of consumer channels. For example, if the consumer node expands the number of consumer channels available to two (\( c = 2 \)), the producer node calculates

\[ pz = \frac{ps}{c} \]

\[ 50 = 100 / 2 \]
The producer node assigns partitions to existing producer channels based on the updated partition size $pz$, and determines an overflow portion of the existing producer channel based on the updated partition size $pz$ and the previous partitions assigned to the producer channel. For example, the producer node dynamically redistributes the partition space as follows:

<table>
<thead>
<tr>
<th>Producer Channel</th>
<th>Partitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>producer channel 0</td>
<td>updated partitions 0-49 ($pz = 50$)</td>
</tr>
<tr>
<td></td>
<td>previous partitions 0-99</td>
</tr>
<tr>
<td></td>
<td>overflow portion partitions 50-99</td>
</tr>
<tr>
<td>producer channel 1</td>
<td>receive overflow portion partitions 50-99</td>
</tr>
</tbody>
</table>

The producer node updates the sharding map accordingly to reflect the redistributed partition space.

[0075] The consumer node may elect to expand further the number of consumer channels available by adding a third consumer channel. The producer node again determines an updated partition size based on an updated count of available consumer channels ($c = 3$):

$$pz = ps / c$$

$$33 = 100 / 3$$

The producer node assigns partitions to existing producer channels based on the updated partition size $pz$, and distributes the overflow portions of the existing producer channels to the new producer channel.

The producer node proceeds to update sharding map 118 using these values for channel partitions map 306 and partition-to-channel index 310.

[0076] In further embodiments, the sharding algorithm assigns partitions to producer channels by determining statistics using channel map 116. For example, the event processing may query the consumer nodes in the cluster to identify available resources for a given consumer node. Non-limiting example resources can include central processing unit (CPU) processor speed or number of cores, available memory or memory speed, available storage capacity or storage read/write speed, network connection speed, a count of existing consumer channels, or combinations thereof. Based on the available resources and on channel map 116, the event processing may determine weighted statistics to assign partitions to producer channels in sharding map 118. For example, if the event processing determines that consumer node U1234567 contains a faster CPU or more memory than other consumer nodes, the event processing may consult channel map 116 to identify consumer channels on consumer node U1234567 (e.g., using a producer node). The sharding algorithm may then assign more partitions to producer channels in sharding map 118 that correspond to consumer channels identified as having more available resources.

[0077] Channel-partitions map 306 tracks a related mapping as partition-to-channel index 310. In some embodiments, channel-partitions map 306 can support elasticity in the cluster, such as horizontal or vertical scaling. Channel-partitions map 306 allows sharding map 118 to identify what partitions are associated with a given producer channel and reassign the associated partitions quickly to expand or contract the cluster. For example, channel-partitions map 306 tracks that producer channel 0 can process partitions 0-32 (index 308a), producer channel 1 can process partitions 50-82 (index 308b), and producer channel 2 can process partitions 33-49 and 83-99 (index 308c).

[0078] Figure 4A illustrates example consumer node 106a for event processing, in accordance with certain embodi-
ments of the present invention. Consumer node 106a includes event listeners 402, consumer event buffers 404, and requestor pool 406. Consumer node 106a uses embedded Hazelcast 126a for communication with other producer nodes and consumer nodes over TCP/IP socket 410.

[0079] In some embodiments, when consumer node 106a comes online and joins the cluster, the consumer node consults its configuration to identify event streams of interest. For example, the configuration may contain a collection of entries from an administrator that list stream types and stream names of event streams for processing by consumer node 106a. Consumer node 106a creates a set of event listeners 402 for each event stream entry in the configuration. Consumer node 106a further creates corresponding consumer event buffers 404. For example, consumer node 106a can create one consumer event buffer per consumer channel in the set of event listeners 402. In some embodiments the number of event listeners 402 in the set is equal to the value defined in the configuration, or defaults to five. Consumer node 106a further creates requestor pool 406 and pairs one requester with each consumer event buffer.

[0080] To create the paired relationship between consumer channels and producer channels, the consumer node registers the newly created consumer channels into the channel map. In some embodiments, the channel map is a Hazelcast distributed map for registering the event listeners and consumer node 106a uses embedded Hazelcast 126a over TCP/IP socket 410 for reading and writing updates to and from the channel map. When one or more producer nodes come online and join the cluster, the producer nodes can then read the channel map to determine how many producer channels to create. In this manner, based on the channel map the producer nodes are able to create and update the sharding map used to transmit events to the various consumer nodes in the cluster.

[0081] Figure 4B further illustrates example consumer node 106a for event processing, in accordance with certain embodiments of the present invention. In some embodiments, consumer node 106a implements each consumer channel in three parts: event receivers 414, consumer event buffers 416, and event consumer executors 418. This architecture improves the ability of consumer node 106a to receive messages with enhanced throughput. With reference to Figure 4A, consumer node 106a uses an event listener registered to receive update notifications from the channel map, for the consumer channel that represents a given event stream. For example, consumer node 106a registers a Hazelcast event listener with the channel map to receive update notifications about an event stream type and stream name of interest.

[0082] The event listener uses a consumer event buffer 416 to buffer received events 412. In some embodiments, consumer event buffers 416 are implemented using ring buffer data structures provided by a Disruptor platform. As those skilled in the art will appreciate, the Disruptor platform refers to a publicly available third-party architecture and processing mechanism for high-speed parallel processing, also referred to as concurrency. In further embodiments the ring buffers can be configured with a single thread input and a single thread output (e.g., using event consumer executors 418). This single thread usage provides enhanced throughput over a traditional usage of the Disruptor platform.

[0083] Regarding traditional usage, the Disruptor architecture provides components that process volumes of data concurrently, with low latency. The Disruptor architecture implements a data structure referred to herein as a ring buffer. A ring buffer performs similarly to a queue. The event processing system can push data onto and pop data off the ring buffer for processing. Disruptor ring buffers can be used similarly to a network of queues that can operate without needing locks. Locks provide thread safety by avoiding unexpected data corruption due to parallel processing, but also add significant and unwanted complexity. Traditionally the Disruptor pattern is designed for a system to use multiple threads to read from and write to ring buffers. However, using the Disruptor architecture in the traditional manner lowers throughput and reduces performance.

[0084] In general, the speed at which the event processing delivers events is affected by the ability to process and serialize the event for delivery. Traditional implementations have projected throughput in the range of 15,000-20,000 events per second. However, this traditional throughput is well below the steady state throughput desired to provide a performant high speed event processing service. To increase throughput, traditional systems can also use multi-threading architectures and aggregate structures. When considering a traditional multi-threading solution, thread contention is one problem to solve. Traditionally, a simplistic approach can use a Java queue with one thread pushing messages onto the queue and one or more threads pulling messages off the queue. To do this, the simplistic approach needs to consider efficient use of both push and pull threads to interact with the ring buffers.

[0085] In contrast to traditional implementations, in the event processing system the single thread input and single thread output from consumer event buffers 416 (and the corresponding producer event buffers) help differentiate consumer node 106a. Furthermore, the paired consumer channel and producer channel and sharding algorithm applied by the producer node help the event processing system increase throughput over fourfold (e.g., 100,000 messages or more per second) when compared with traditional implementations.

[0086] The consumer node passes the received events to event consumer executors 418. Some embodiments of event consumer executors 418 identify a rule associated with received event 412. For example, event transport metadata that the producer node bundles with the event can help identify the appropriate rule for consumer node 106a. Consumer
node 106a can then make a call 420 to execute the corresponding rule. For example, event consumer executors 418 can call into the rules engine to get a pre-allocated requestor object dedicated to execution of the rule associated with the received event.

**[0087]** Figure 5A illustrates example producer node 104a for event processing, in accordance with certain embodiments of the present invention. Producer node 104a includes event stream 502, event sharding module 504, and channel pool 506. Producer node 104a is in communication with other producer and consumer nodes in the cluster using embedded Hazelcast 124a and TCP/IP socket 510.

**[0088]** An event service can begin to transmit events to producer node 104a via event streams 502. In some embodiments, the event processing helps the events to arrive in the cluster in the order in which the event service presented the events. When producer node 104a comes online and joins the cluster, producer node 104a retrieves and reads the channel map from the cluster. In some embodiments, the channel map is a Hazelcast distributed map and producer node 104a uses embedded Hazelcast 124a to read the channel map over TCP/IP socket 510. Producer node 104a uses the channel map to determine updates to the sharding map and identify what producer channels to create. The channel map contains information about the consumer nodes in the cluster and which event streams and consumer channels the consumer nodes support. In some embodiments, producer node 104a uses the sharding map in combination with the channel map to determine the consumer node destination of a given event. For example, producer node 104a first receives event stream 502. Producer node 104a shards, or partitions, event stream 502 into individual events (e.g., using event sharding module 504). After sharding event stream 502into individual events, producer node 104a determines a sharding key for each event based on an event key generated upon receiving the event (e.g., also using event sharding module 504). Producer node 104a looks up the corresponding producer channel among channel pool 506, based on the sharding key in the sharding map. The sharding map identifies a producer channel for transmitting the event. The sharding map helps producer node 104a ultimately deliver a given event consistently to the same producer channel that corresponds to the sharding key and partition determined by the producer node. The paired relationship between the producer channel and consumer channel consequently help the producer node ultimately deliver the event to the same consumer channel on the same consumer node for a given event stream.

**[0089]** Figure 5B further illustrates example producer node 104a for event processing, in accordance with certain embodiments of the present invention. Upon coming online and joining the cluster, producer node 104a creates producer channels in conjunction with the universe of consumer channels available in the cluster, as tracked in the channel map based on event streams (e.g., based on stream type and stream name). When a remote client requests event publisher 514, producer node 104a receives from the remote client the event stream to which the remote client would like to publish events. For example, producer node 104a can allocate producer channels based on stream type and stream name. The channel map tracks for producer node 104 how many consumer channels are available for a given event stream, for each consumer node in the cluster. In some embodiments, event publisher 514 then creates a corresponding set of producer event buffers 518. For example, event publisher 514 can produce a one-to-one match of producer event buffers on the producer node in a paired relationship with a corresponding event listener on a consumer node in the cluster.

**[0090]** Producer node 104a shards 516 the received event stream into individual events according to the following algorithm. Producer node 104a divides the received event stream into individual events. For a given event 512, producer node 104a determines an event key for the event. In some embodiments, the event key is determined based on the contents of event 512. For example, the event key may be a hash code or fingerprint of the contents of event 512. Producer node 104a computes the sharding key by determining a modulo of the event key with the partition space. Producer node 104a identifies a producer channel for transmitting the event, by retrieving the corresponding sharding key in the sharding map for the event stream. Event publisher 514 places event 512 into producer event buffer 518 in the producer channel identified by the sharding map for the event stream. Producer event buffer 518 then routes event 512 from the producer channel to the consumer channel on the corresponding consumer node in the cluster. Upon receiving event 512, the consumer node buffers event 512 for execution.

**[0091]** Figure 6 illustrates example event processing system 600, in accordance with certain embodiments of the present invention. Event processing system 600 includes event reader 602 in communication with producer node 104a, which is in communication with consumer node 106a. Event processing system 600 is able to scale to an improved throughput of hundreds of thousands of messages per second. In contrast, traditional message bus architectures are only able to achieve example throughputs of 15,000-25,000 messages per second. To achieve this enhanced throughput, event processing system 600 employs a combination of technology configured in a unique way.

**[0092]** One aspect that contributes to enhanced throughput is a symbiotic paired relationship between producer channels 314a-c and consumer channels 608a-c. Specifically, some embodiments of the event processing system leverage a symbiotic paired relationship between producer channels 314a-c and producer event buffers 518 on producer node 104a, and consumer channels 608a-c and consumer event buffers 416 on consumer node 106a. Event processing system 600 achieves massive speed because producer node 104a separates and buffers received events so that a single producer event buffer 518 has a direct TCP/IP connection to a corresponding single consumer channel 608a-c and consumer event buffer 416. In some embodiments, this symbiotic paired relationship also allows event processing...
system 600 to include flow control per paired producer channel 314a-c and consumer channel 608a-c. For example, event processing system 600 can affect the rate at which a consumer channel 608a-c receives and processes events, and match or modulate the speed on a producer event buffer 604a to achieve the desired rate indirectly on a corresponding consumer channel 608a-c. Further embodiments of the flow control can be dynamic, for example by altering the flow rate periodically (e.g., about every thousand messages). The result of this flow control is a steady state in which a producer event buffer 518 and a consumer event buffer 416 on a corresponding consumer channel 608a-c are operating at a related rate to achieve enhanced throughput.

Event processing system 600 begins by receiving an event stream from event service 102a. For example, an external source of the event stream can be a Twitter feed. Some embodiments of event processing system 600 can use an application programming interface (API) for retrieving social media data to retrieve the event stream of interest, such as an API provided by Gnip, Inc. of Boulder, Colorado, United States. Event processing system 600 illustrates a cluster including producer node 104a in communication with event service 102a and consumer node 106a. Consumer node 106a announces an ability to accept event streams for Twitter events for Dept.A. In this example an administrator has configured consumer node 106a to have three active consumer channels (channel 0 (608a), channel 1 (608b), channel 2 (608c)) for event stream Twitter/Dept.A. Consumer channels 0-2 (608a-c) receive and process events from producer node 104a. Producer node 104a is in communication with event service 102a serving the event stream for Twitter/Dept. A in the cluster.

Consumer node 106a contains three event receivers 414 associated with consumer channels 0-2 (608a-c). Each event receiver 414 has an associated consumer event buffer 416 along with an event consumer executor 418. Event consumer executor 418 pops events off consumer event buffers 416 for execution. In some embodiments, consumer event buffers 416 can be ring buffers using the Disruptor architecture.

Event processing system 600 illustrates symbiotic paired relationships on producer node 104a. Producer node 104a contains a producer channel 0-2 (314a-c) and producer event buffer 518 for each corresponding consumer channel 0-2 (608a-c). Producer event buffers 518 are in a paired relationship with each corresponding consumer channel 0-2 (608a-c). Each producer ring buffer 518 has a producer mechanism 520 that pops events off producer event buffers 518 for transmission to the corresponding consumer channel 0-2 (608a-c).

Event publisher 514 contains sharding logic 516 to shard, or partition, the received event stream into individual events including event 610. Event publisher 514 directs the sharded event 610 to one of producer channels 314a-c which directs the event in turn to an associated producer event buffer 518. Event publisher 514 determines the desired producer channel 314a-c using the sharding map based on a sharding key for event 610.

Figure 6 also illustrates an example path taken by an individual event 610. Event reader 602 begins by reading and sharding event 610 from an event stream provided by event service 102a (step E-1). For example, event service 102a can be a Twitter feed. Event 610 can have contents “Hello world” corresponding to an event key of 190. Producer node 104a determines a sharding key and corresponding producer channel for the event (step E-2). In some embodiments, producer node 104a determines the sharding key based on the event key and the partition space. For example, producer node 104a can compute a modulo of the event key with the partition space. If the partition space has a value of 100, then 190 % 100 = 90, where x % y represents a modulo operator that returns the remainder of x divided by y. The resulting sharding key is therefore 90. With reference to Figure 3B, looking up the sharding key of 90 in partition-to-channel index 312c of sharding map 118 yields producer channel 2 (314c).

Producer node 104a passes the sharded event to producer channel 2 (314c) which inserts the event into the associated producer event buffer (step E-3). The event winds its way through the producer event buffer. An extraction thread in the associated event producer extracts the event (step E-4). The extraction thread serializes the event and transmits the event to consumer channel 2 (608c), for example using TCP/IP (step E-5). The receiver associated with consumer channel 2 (608c) receives the event (step E-6) and inserts the received event into the consumer event buffer associated with consumer channel 2 (608c). The consumer event buffer buffers the received event waiting to be processed (step E-7). The event makes its way around the consumer event buffer and the associated event consumer executor retrieves the event using an extraction thread for execution (step E-8). Consumer node 106a executes processing associated with the event (step E-9). In some embodiments, consumer node 106a converts the event using internal data structures and uses event transport metadata bundled with the event to identify rules of interest for processing the event using an associated rule engine to execute the rules. Consumer node 106a then makes call 420 to execute the identified rule. In some embodiments, the identified rule can be an activity rule that identifies an activity for event processing system 600 to perform.

Figure 7 illustrates example event processing system 700, in accordance with certain embodiments of the present invention. Event processing system 700 illustrates sharding events that have different event keys and sharding keys. Producer node 104a receives an incoming event stream 702 that includes multiple events. Each event has an event key associated with the event, illustrated as E-9, E-25, E-46, E-109, etc. Some embodiments of event processing system 700 can determine the event key based on the contents of the event, such as a hash code or fingerprint of the event. Event reader 602 reads event stream 702 from an event service (not shown) and converts the event stream into...
an internal event form that event processing system 700 can process. Event publisher 514 shards 516 event stream 702 into individual events. Each event has the event key illustrated (e.g., E-9, E-25, E-46, E-109, etc.). In some embodiments, producer node 104a preserves the order of arrival of the individual events in event stream 702.

Producer node 104a uses the sharding map and partition space to determine a sharding key for a given event. The sharding key identifies the producer channel 314a-c to receive the sharded event. In some embodiments, the sharding algorithm of computing a modulo of the event key with the partition space determines the sharding key. For example, given an event key of E-109 and a partition space of 100, one embodiment of the sharding algorithm would return a sharding key of 109 % 100 = 9. Producer 104a can look up the sharding key in the sharding map to identify the appropriate producer channel 314a-c for the event. With reference to Figure 3B, for a sharding key of 9, partition-to-channel index 312a of sharding map 118 identifies producer channel 0 (314a).

Event publisher 514 provides the sharded events to producer channels 0-2 (314a-c) in order, in groups 704a-c (also illustrated in white, black, and gray). Producer channels 0-2 (314a-c) use producer event buffers 518 and event producers 520 to transmit the sharded events. The sharded events travel over TCP/IP to corresponding consumer channels 0-2 (608a-c) on consumer node 106a. Event receivers 414 receive the sharded events into consumer event buffers 416. Event consumer executors 418 then make calls 420 to execute any rules identified by event transport metadata bundled with the received events. Consumer node 106a may execute the identified rules on a rules engine that is remote to consumer node 106a and producer node 104a, or the rules engine may be locally installed on the consumer node or producer node.

Example use cases

GoCars Corp may process large volumes of social media events with enhanced throughput using event processing system 800. GoCars Corp is launching a new marketing campaign that introduces a new type of navigation system for an existing line of cars called "GoCars." GoCars Corp would like to analyze events such as Twitter messages to see what people are saying about its new offering.

Producer node 104a begins by making a connection to event service 102a. For example, event service 102a may be a social media feed such as a Twitter feed. Producer node 104a proceeds to retrieve events from event service 102a. In some embodiments producer node 104a determines the appropriate consumer node for processing the events based on parsing a subject of the event to identify a predetermined string, and based on a location of the author of the social media post. For example, producer node 104a may parse the event to detect the text "GoCars."

If producer node 104a identifies the event to be of interest (e.g., if the event contains the predetermined string), producer node 104a determines the sharding key for the event based on a location of the author of the social media post. For example, producer node 104a may determine a zip code associated with the author (e.g., by aggregating zip code data from an external system). Event processing system 800 may configure the channel map and sharding map to assign consumer channels and consumer nodes based on zip code. For example, event processing system 800 may use embedded Hazelcast 124a, 126a-b to configure the channel map and sharding map to transmit events associated with even zip codes to consumer channels on consumer node 106a and odd zip codes to consumer channels on consumer node 106b. In further embodiments, event transport metadata bundled with the event may contain information on how consumer nodes 106a-b should process the event when received.

After receiving the event, producer node 104a determines an associated consumer channel for the event based on the sharding map and on a sharding key for the event. In this example, the sharding map associates the event with a consumer channel on consumer node 106a, based on the zip code. Producer node 104a provides the received event to a producer event buffer in a producer channel. The producer channel uses TCP/IP socket connection 510 to transmit the received event directly to a consumer event buffer associated with a corresponding consumer channel on consumer node 106a. Consumer node 106a receives the event via TCP/IP socket 410 and the consumer event buffer. Consumer node 106a unpacks the event and executes a rule identified by the event transport metadata. In some embodiments, the event transport metadata may identify a rule that causes consumer node 106a to perform sentiment analysis on the received event. Sentiment analysis can use machine learning and decisioning techniques in conjunction with analytic models to identify positive or negative sentiment in the contents of the received event. The event transport metadata may identify a further rule that causes consumer node 106a to store the event to persistent storage, based on the positive
or negative result of the sentiment analysis. For example, consumer node 106a may store events with positive sentiment
into a marketing database for customer highlights, and events with negative sentiment into a customer database for
customer service follow-ups.

[0107] Figure 9 illustrates example event processing system 900 for high speed record processing, in accordance
with certain embodiments of the present invention.

[0108] In this example, a customer has an external source system that provides line item details for invoices (e.g.,
medical claims for health insurance). An administrator deploys a rules-based application into a cluster. Consumer nodes
106a-b collate and compute aggregate information about item detail records associated with line items. The event service
includes high speed database 902 capable of producing result sets that can be iterated at speeds in the range of 100,000
rows per second. In this application, an incoming record may have an overall case number that groups individual item
detail records together. Item detail records may also have a field or indicator that tracks "end sub-group record," for
example. Upon receiving an item detail record with its "end sub-group record" indicator set to true, the application may
be configured to collect or group item detail records and write out the set as a single record, for example to database 904.

[0109] Producer node 104a connects to an event service such as database 902. Some embodiments of producer
node 104a use an event publisher to retrieve events, for example using a database specific query. Producer node 104a creates
an event for each row retrieved from database 902 using the database-specific query. In some embodiments, producer
node 104a determines the sharding key for each event based on an identified field from the database. For
example, the sharding key in conjunction with the sharding map may determine that producer node 104a should route
given event to consumer node 106a. The sharding map directs the event to a producer channel corresponding to a
consumer node associated with consumer node 106a. Producer node 104a uses TCP/IP socket connection 510 to
transmit the event directly to consumer node 106a.

[0110] Consumer node 106a receives the event via TCP/IP socket 410 into its consumer event buffer from producer
node 104a. In some embodiments, consumer node 106a unpacks the received event including bundled event transport
metadata. For example, the event transport metadata may identify a rule that checks whether the received event is part
of an "existing" set or a "new" set in database 904. Consumer node 106a executes the rule identified by the event
transport metadata. The event transport metadata may further identify or define rules creating a structure containing a
"new" bucket, or retrieving an "existing" bucket based on the event ID or case ID. In further embodiments, if the event
has its "end sub-group" record configured to true, event processing system 900 may group together the events in the
bucket in a common format and store the grouped events in database 904 to allow further analysis and processing,
potentially by follow-on applications.

[0111] Figure 10 illustrates example event processing system 1000 for performing targeted actions while aggregating
external system input, in accordance with certain embodiments of the present invention. Event system 1000 includes
producer node 104a in communication with event service 102a and consumer nodes 106a-b.

[0112] In the illustrated example, a company has deployed an external source system that pushes information required
by an customer service representative who is online. The company may have a call center where customer service
representatives use a rules-based application in communication with event processing system 1000 to respond to end
users who call in with questions and answer those questions. In this call center, the application notifies a customer
service representative that a call has come in that the representative will respond to, and routes the call to the repre-
sentative. In addition, the application sends a query to an external system requesting additional information about the
incoming caller. If the external system identifies information for consideration by the representative, the external system
delivers the information to an entry point (e.g., using event service 102a). The delivered information requires routing to
the correct representative. In the cluster illustrated in Figure 10, the representatives may be deployed on a diverse set
of servers (e.g., one or more of consumer nodes 106a-b). Event processing system 1000 locates the physical consumer
node on which a representative is logged in who should receive the packaged information, packages up the information
from the external system, and sends the packaged information to the correct consumer node 106a. Consumer node
106a accepts the event containing the packaged information and forwards the packaged information to the representative.
For example, the application may display the packaged information in a pop-up window on a web browser on the
representative's computer 1002.

[0113] The customer service representative begins by using the application deployed by the company on computer
1002. For example, the representative may initiate a new case to track an issue faced by a caller. In some embodiments,
event processing system 1000 may present a user interface for a customer service application on computer 1002, in
communication with a rules engine and rules base (not shown). The representative on computer 1002 may be assigned
to consumer node 106a. In some embodiments, the rules engine and rules base may be deployed on one or more
remote systems in communication with producer node 104a or consumer nodes 106a-b. In other embodiments, the rules
engine and rules base may be deployed locally on producer node 104a or consumer nodes 106a-b.

[0114] In connection with the call received by the representative on consumer node 106a, producer node 104a receives
information from event service 102a. For example, event service 102a may be an external information service that
provides information about the caller. Producer node may create an event that packages the information from event
service 102a. Event processing system 1000 may send a query over TCP/IP socket connection 510 to identify the consumer node associated with the representative helping the caller. The query results may identify the representative and associated customer node 106a.

In some embodiments, producer node 104a may use embedded Hazelcast client 124a to update the sharding map to reflect that received events associated with the caller and representative will route to consumer node 106a. In some embodiments, the representative and caller ID information for the caller may be included as event metadata. Accordingly, when producer node 104a receives the event from event service 102a, producer node 104a provides the event to the correct producer channel based on the sharding map, which transmits the event over TCP/IP socket connection 510 directly to a consumer channel on consumer node 106a as intended. The corresponding consumer event buffer on consumer node 106a receives the event over TCP/IP socket 410. Consumer node 106a processes the event and reads the bundled event transport metadata. The event transport metadata may identify or define a rule to notify the representative on computer 1002 of the information from the external information service. In some embodiments, the application may notify the representative by displaying a pop-up window in a web browser on computer 1002.

Elasticity

Figure 11 illustrates an example event processing method 1100 for providing elasticity, in accordance with certain embodiments of the present invention. Method 1100 supports elasticity such as horizontal and vertical scaling. The elasticity techniques described herein including method 1100 allow the event processing system to reconfigure the cluster dynamically while the cluster is running. Horizontal scaling refers to support by the event processing for dynamically adding or removing producer and consumer nodes to or from the cluster. Vertical scaling refers to support for dynamically adding or removing consumer channels to or from an individual consumer node, or producer channels to or from an individual producer node.

For example, method 1100 supports adding a new consumer node to the cluster. When the new consumer node comes online and joins the cluster, the new consumer node begins by registering in a consumer map, and updates the channel map to reflect the event streams and corresponding consumer channels that the new consumer node supports (step 1110). In some embodiments, the consumer map may be a distributed Hazelcast map. When registering in the consumer map, the new consumer node may register initially with a status of Pending.

In response to detecting updates to the channel map, the producer nodes update their configurations (step 1120). For example, the producer nodes update their producer channels based on the updated consumer channel configuration in the channel map. The producer nodes also update the sharding map to assign sharding keys to the updated producer channels. Finally, the producer nodes proceed to buffer events for transmission over the updated producer channels.

In response to detecting updates to the sharding map, all consumer nodes in the cluster determine a consumer-node-wide delta reflecting data to be moved, and copy the data to be moved into a cluster change data map (step 1130). In some embodiments, the cluster change data map may be a distributed Hazelcast map. In response to detecting updates to the cluster change data map, the new consumer node copies the moved data, clears the moved data from the cluster change data map and from the consumer node that previously stored the data, and updates the status to Active for the new consumer node in the consumer map (step 1140). In response to detecting status updates in the consumer map, the producer nodes begin providing data to the producer channel for transmission to the corresponding consumer channels on the new consumer node (step 1150).

Figure 12 illustrates an example event processing system 1200 expanding to add new consumer node 1202, in accordance with certain embodiments of the present invention. Event processing system 1200 includes producer node 104a in communication with consumer nodes 106a-b, 1202 using consumer map 1206, producer map 1208, sharding map 118, and cluster change data map 1210.

Event processing system 1200 supports horizontal scalability by expanding dynamically upon addition of a new consumer node. The addition of consumer node 1202 to the cluster can distribute and potentially increase the event processing capacity of the cluster. The process of adding consumer node 1202 involves a number of changes and updates in the cluster. In some embodiments, there are two sets of changes: one set for consumer nodes and a companion set for producer nodes. Although the changes are described sequentially, in some embodiments some steps can happen in parallel and in different threads, to streamline the addition of consumer node 1202.

New consumer node 1202 begins by coming online and joining the cluster. In general, as part of the startup process new consumer node 1202 orchestrates the rebalancing of the cluster and identifies itself as ready when the process is complete. Consumer node 1202 continues by configuring event stream support. For example, consumer node 1202 reads its configuration to identify supported event streams by stream type and stream name. Consumer node 1202 proceeds to initiate a cluster lock for re-sharding. In some embodiments, the cluster lock on re-sharding prevents initiation of additional re-sharding sessions while the current session is in progress. Consumer node 1202 then registers itself with consumer map 1206. Consumer map 1206 tracks the universe of consumer nodes in the cluster, and producer
map 1208 tracks the universe of producer nodes in the cluster. In some embodiments, consumer node 1202 registers with initial status 1212 of Pending.

Based on the event stream support, consumer node 1202 creates corresponding consumer channels and consumer ring buffers to reflect the supported event streams. Consumer node 1202 determines updates to the channel map based on the consumer channels. Consumer node 1202 then initiates a wait loop, waiting for related activity on other consumer and producer nodes in the cluster to signal finalization. In some embodiments, consumer node 1202 registers event listeners or item listeners on cluster change data map 1210 to receive update notifications for changes to the cluster change data map.

Each producer node 104a reacts to changes in the channel map by new consumer node 1202. In some embodiments, producer node 104a reacts by transmitting an end event on all existing producer channels to all corresponding consumer channels in communication with producer node 104a. Producer node 104a proceeds to update the producer channels. For example, producer node 104a creates and initializes new producer event buffers corresponding to the updated producer channels. Producer node 104a next updates the sharding map to reflect the updated producer channels. In some embodiments, producer node 104a pauses production of events to copy the new sharding map, and resumes production of events based on the new sharding map. For events targeted to new consumer node 1202, producer node 104a buffers those events until status 1212 of consumer node 1202 becomes Active in consumer map 1206.

After producer node 104a provides an end event on all producer channels for transmission to all consumer channels, consumer nodes 106a-b, 1202 corresponding to the consumer channels process the received end event. Although the processing of the end event is described herein with reference to consumer node 106a, the processing is the same for all consumer nodes in the cluster. For example, consumer node 106a triggers all consumer event buffers corresponding to all consumer channels to process the received end event as follows. Consumer node 106a reads the updated channel map, and creates a consumer-node-wide delta reflecting the updates. As each consumer event buffer receives the end event to process, the consumer event buffer copies its data to be moved to cluster change data map 1210. Once the consumer event buffer has completed copying the data to move, the consumer event buffer resumes processing any events from the corresponding producer event buffer that have been buffered.

In response to a notification of updates to cluster change data map 1210, new consumer node 1202 continues to wait for all moved partitions to appear in cluster change data map 1210. Once the moved partitions all appear in cluster change data map 1210, consumer node 1202 copies the data from cluster change data map 1210 and clears the contents it copied from cluster change data map 1210 and from the consumer nodes that previously stored the copied data. Upon completion of copying the data, consumer node 1202 updates its status to Active in consumer map 1206 and clears the cluster lock on re-sharding. Lastly, upon receiving a notification of the status update for consumer node 1202 in consumer map 1206, producer node 104a begins providing events to the producer channels associated with consumer node 1202, and the producer channels begin transmitting the events directly to consumer node 1202.

The elasticity support in event processing system 1200 has thus far discussed adding a consumer node to the cluster. Method 1200 also supports the following dynamic rebalancing: removing consumer node 1202, adding consumer channels to a consumer node in support of a given event stream, and removing consumer channels from a consumer node for the event stream.

With reference to Figure 11, in some embodiments event processing system 1200 supports removing a consumer node from the cluster in support of horizontal scalability. Dynamic rebalancing in response to deleting consumer node 1202 can be similar to adding a consumer node. The difference is that the updates to the channel map (step 1110) and sharding map (step 1120) include removing the appropriate entries in the channel map and sharding map, rather than adding new entries. With that change in mind, method 1100 proceeds as described earlier in response to producer node 104a and consumer nodes 106a-b, 1202 receiving notifications of the updates to the channel map and sharding map. That is, in response to the removed consumer channels from the channel map, producer node 104a removes producer channels corresponding to the consumer channels that will be removed when consumer node 1202 is removed from the cluster, and updates the sharding map to reflect the updated producer channel configuration (step 1120). In response to detecting updates to the sharding map, consumer node 1202 determines a delta that reflects data previously on consumer node 1202 to be copied to the cluster change data map and moved to remaining consumer nodes 106a-b (step 1130). In response to detecting the updates to the cluster change data map, remaining consumer nodes 106a-b copy the data to be moved from consumer node 1202, clear the moved data from the cluster change data map once copied, and consumer node 1202 updates its status in the consumer map to reflect its removal from the cluster (step 1140). In response to detecting the status update in the consumer map, producer node 104a proceeds to provide events for transmission to producer channels, for transmission to the updated consumer channels on remaining consumer nodes 106a-b (step 1150).

In some embodiments, event processing system 1200 supports adding consumer channels to an existing consumer node in support of vertical scalability. In contrast to adding a new consumer node, the update to the channel map comprises changing (i.e., increasing) the number of consumer channels supported for a given event stream in an existing updated consumer node. Upon receiving a notification of an update to the channel map, event processing system
1200 rebalances the cluster by processing the channel map and sharding map in a similar manner as for addition of a new consumer node. That is, the updated consumer node updates the channel map to reflect the increased consumer channels for event streams of interest (step 1110). In response to detecting updates to the channel map, producer node 104a creates producer channels corresponding to the new consumer channels created on the existing consumer node, and updates the sharding map to reflect the new producer channels (step 1120). In response to detecting updates to the sharding map, the consumer node determines a delta that reflects data previously assigned to other consumer channels or other consumer nodes, and copies the data to be moved into the cluster change map (step 1130). In response to detecting the updates to the cluster change map, the existing updated consumer node copies the moved data and clears the moved data from the cluster change map and from the previous consumer node storing the data prior to moving (step 1140). After copying the moved data, producer node 104a begins providing data to the new producer channels for transmission to the updated consumer channels (step 1150).

[0130] In some embodiments, event processing system 1200 supports removing consumer channels from an existing consumer node. In this case the update to the channel map comprises changing (i.e., decreasing) the number of consumer channels supported for a given event stream. Event processing system 1200 processes the updates to the channel map in a similar manner as with removal of an existing consumer node. That is, the updated consumer node updates the channel map to reflect the decreased consumer channels for event streams of interest (step 1110). In response to detecting updates to the channel map, producer node 104a removes producer channels corresponding to the removed consumer channels created on the existing consumer node, and updates the sharding map to reflect the removed producer channels (step 1120). In response to detecting updates to the sharding map, the consumer node determines a delta that reflects data to be assigned to other consumer channels on the consumer node or on other consumer nodes, and copies the data to be moved into the cluster change map (step 1130). In response to detecting the updates to the cluster change map, the existing updated consumer node copies the moved data and clears the moved data from the cluster change map and from the previous consumer node that stored the data prior to moving (step 1140). After copying the moved data, producer node 104a begins providing data to the new producer channels for transmission to the updated consumer channels (step 1150).

[0131] The event processing systems and methods described herein address a technical problem of rapidly routing and processing large volumes of discrete events in a networked cluster. The event processing provides a technical contribution that involves determining a sharding key for a received event from an event stream, and using a sharding map to correlate the sharding key for the event with a producer channel. A producer node provides the received event to a producer event buffer associated with the producer channel, and the producer event buffer transmits the event to a corresponding consumer event buffer associated with a consumer channel on a consumer node. The sharding map, sharding key, and paired producer channels and consumer channels, among other aspects, add specific limitations other than what is well-understood, routine, and conventional in the field. The event processing further provides significantly more than an abstract idea by improving the functioning of the producer and consumer digital data processors themselves. For example, the event processing achieves improved throughput in the cluster by leveraging the paired relationship between producer channels on the producer node and consumer channels on the consumer node. The event processing also supports dynamic rebalancing of the system in response to adding or removing producer or consumer nodes, or adding or removing producer or consumer channels to or from producer or consumer nodes. These enhancements elevate the producer and consumer digital data processors beyond their conventional functioning.

[0132] Other embodiments are within the scope and spirit of the event processing systems and methods. For example, the event processing functionality described above can be implemented using software, hardware, firmware, hardwiring, or combinations of any of these. One or more digital data processors operating in accordance with instructions may implement the functions associated with event processing in accordance with the present disclosure as described above. If such is the case, it is within the scope of the event processing systems and methods that such instructions may be stored on one or more non-transitory computer-readable storage media (e.g., a magnetic disk, solid state drive, or other storage medium). Additionally, as described earlier, modules implementing functions may also be physically located at various positions, including being distributed such that portions of functions are implemented at different physical locations.

[0133] The event processing systems and methods are not to be limited in scope by the specific embodiments described herein. Indeed, other various embodiments of and modifications to the event processing, in addition to those described herein, will be apparent to those of ordinary skill in the art from the foregoing description and accompanying drawings. Thus, such other embodiments and modifications are intended to fall within the scope of the event processing systems and methods described herein. Furthermore, although the event processing has been described herein in the context of a particular implementation in a particular environment for a particular purpose, those of ordinary skill in the art will recognize that its usefulness is not limited thereto and that the event processing may be beneficially implemented in any number of environments for any number of purposes.
Claims

1. A digital data processing system comprising
   a producer node in communicative coupling with one or more consumer nodes and with a sharding map, wherein
   the producer node is configured to:

   receive at least one event stream comprising a plurality of events;
   determine a sharding key associated with an event among the plurality of events in the event stream;
   identify, based on the sharding map, a producer event buffer associated with a producer channel on the producer
   node for transmitting the event to a corresponding consumer event buffer associated with a consumer channel
   on a consumer node among the one or more consumer nodes, wherein the sharding map correlates the sharding
   key for the event with the producer channel; and
   provide the event to the producer event buffer associated with the producer channel in order to transmit the
   event to the corresponding consumer event buffer associated with the consumer channel on the consumer node.

2. The system of claim 1, wherein the producer node is further configured to initialize a plurality of producer channels by
   referencing a channel map that correlates an event in the event stream with one or more consumer channels on
   the one or more consumer nodes;
   creating at least one producer channel based on the channel map that is communicatively coupled with a corre-
   sponding consumer channel among the one or more consumer channels; and
   updating the sharding map to correlate the sharding key for the event with the at least one producer channel created
   based on the channel map.

3. The system of claim 1 or 2,
   wherein the sharding map correlates the sharding key for the event with the producer channel based on a partition
   space, and
   wherein the partition space is determined using a partition criterion based on a count of consumer nodes available
   to process the plurality of events in the event stream.

4. The system of claim 3, wherein the producer node is further configured to update the sharding map in response to
   detecting an update to a channel map that correlates an event in the event stream with one or more consumer
   channels on the one or more consumer nodes; in which case optionally:

   a) wherein the system is configured to update the sharding map by redistributing the partition space based on
      determining an updated partition size for existing producer channels based on a count of available consumer
      channels, and
      assigning an overflow portion of the partition space based on the updated partition size to a new producer
      channel; or
   b) wherein the producer node is configured to update the sharding map by redistributing the partition space
      based on
      querying the one or more consumer nodes to identify resources available to the one or more consumer nodes, and
      weighting the partitions assigned to each producer channel based on the identified resources.

5. The system of any of claims 1 to 4, wherein the producer node is any of:

   a) further configured to adjust the rate of providing events to the producer event buffer so as to manipulate the
      rate of events processed on the corresponding consumer node by the consumer event buffer, in order to improve
      throughput between the producer node and the consumer node; and
   b) further configured to bundle event transport metadata with the plurality of events in the event stream for
      transmission to the consumer node, wherein the event transport metadata contains instructions for identifying
      a rule to process the plurality of events after receipt by the consumer node, and
      trigger the consumer node to execute the identified rule for processing the plurality of events.

6. The system of any of claims 1 to 5, wherein the producer node is further configured to update a plurality of producer
   channels in response to detecting an update to at least one of the sharding map and a channel map by the consumer
   node,
   wherein the update to the at least one of the sharding map and the channel map comprises at least one of adding
   a consumer node to the system, removing a consumer node from the system, adding one or more consumer channels

21
to the consumer node, and removing one or more consumer channels from the consumer node, and
wherein the channel map correlates an event in the event stream with one or more consumer channels on the one
or more consumer nodes.

7. The system of claim 6,
wherein the producer node is further configured to transmit a message to the one or more consumer nodes in
response to detecting the update to at least one of the sharding map and the channel map, and
wherein the transmitted message triggers the one or more consumer nodes to:

determine a delta that tracks the update to the at least one of the sharding map and the channel map,
identify data to be moved to at least one of a different consumer channel and a different consumer node, and
copy the data to be moved to a cluster change data map, wherein the data copied to the cluster change data map
triggers the consumer node to
copy the moved data from the cluster change data map,
clear the copied data from the cluster change data map, and
update a status of the consumer node to active in a consumer map that tracks the one or more consumer nodes
in the cluster,

wherein the producer node is further configured to resume transmission of data to the consumer node in response
to detecting the updated status of the consumer node in the consumer map.

8. A method for event processing, the method comprising:
receiving, by a producer node, at least one event stream comprising a plurality of events, wherein the producer
node is in communicative coupling with one or more consumer nodes and with a sharding map;
determining, with the producer node, a sharding key associated with an event among the plurality of events in
the event stream;
identifying, based on the sharding map, a producer event buffer associated with a producer channel on the
producer node for transmitting the event to a corresponding consumer event buffer associated with a consumer
channel on a consumer node among the one or more consumer nodes, wherein the sharding map correlates
the sharding key for the event with the producer channel; and
providing the event to the producer event buffer associated with the producer channel in order to transmit the
event to the corresponding consumer event buffer associated with the consumer channel on the consumer node.

9. The method of claim 8, further comprising initializing a plurality of producer channels by:
referencing, with the producer node, a channel map that correlates an event in the event stream with one or
more consumer channels on the one or more consumer nodes;
creating at least one producer channel based on the channel map that is communicatively coupled with a
corresponding consumer channel among the one or more consumer channels; and
updating the sharding map to correlate the sharding key for the event with the at least one producer channel
created based on the channel map.

10. The method of claim 8 or 9,
wherein the sharding map correlates the sharding key for the event with the producer channel based on a partition
space, and
wherein the partition space is determined using a partition criterion based on a count of consumer nodes available
to process the plurality of events in the event stream.

11. The method of claim 10, further comprising updating, with the producer node, the sharding map in response to
detecting an update to a channel map that correlates an event in the event stream with one or more consumer
channels on the one or more consumer nodes, in which case optionally:
a) wherein the updating the sharding map comprises redistributing the partition space by:

determining, with the producer node, an updated partition size for existing producer channels based on a
count of available consumer channels, and
assigning an overflow portion of the partition space based on the updated partition size to a new producer channel; or

b) wherein the updating the sharding map comprises redistributing the partition space by:

- querying the one or more consumer nodes to identify resources available to the one or more consumer nodes, and
- weighting the partitions assigned to each producer channel based on the identified resources.

12. The method of any of claims 8 to 11, further comprising adjusting, with the producer node, the rate of providing events to the producer event buffer so as to manipulate the rate of events processed on the corresponding consumer node by the consumer event buffer, in order to improve throughput between the producer node and the consumer node.

13. The method of any of claims 8 to 12, further comprising:

b) bundling, using the producer node, event transport metadata with the plurality of events in the event stream for transmission to the consumer node, wherein the event transport metadata contains instructions for identifying a rule to process the plurality of events after receipt by the consumer node, and

triggering the consumer node to execute the identified rule for processing the plurality of events.

14. The method of any of claims 8 to 13, further comprising updating a plurality of producer channels in response to detecting an update to at least one of the sharding map and a channel map by the consumer node, wherein the update to the at least one of the sharding map and the channel map comprises at least one of adding a consumer node to the system, removing a consumer node from the system, adding one or more consumer channels to the consumer node, and removing one or more consumer channels from the consumer node, and wherein the channel map correlates an event in the event stream with one or more consumer channels on the one or more consumer nodes.

15. The method of claim 14, further comprising:

transmitting, using the producer node, a message to the one or more consumer nodes in response to detecting an the update to at least one of the sharding map and the topic channel map,

wherein the transmitted message triggers the one or more consumer nodes to:

- determine a delta that tracks the update to the at least one of the sharding map and the topic channel map,
- identify data to be moved to at least one of a different consumer channel and a different consumer node, and
- copy the data to be moved to a cluster change data map,

wherein the data copied to the cluster change data map triggers the consumer node to

- copy the moved data from the cluster change data map,
- clear the copied data from the cluster change data map, and
- update a status of the consumer node to active in a consumer map that tracks the one or more consumer nodes in the cluster, and

resuming transmission, using the producer node, of data to the consumer node in response to detecting the updated status of the consumer node in the consumer map.
FIG. 2A

Consumer Nodes 106a-c

Receive events
Data: Contents
Metadata: Author Stream name Stream type Event transport metadata

Event processor: Processes received events based on event transport metadata

Producer Nodes 104a-b

Shard into events
Data: Contents
Metadata: Author Stream name Stream type Event transport metadata

Event processor determines: Sharding key Producer channel Rate of providing events to producer channel

Event services 102a-c

Serve event stream
Data: Contents
Metadata: Author Stream name Stream type
Receive event stream

Determine sharding key for event

Identify producer event buffer associated with producer channel based on sharding key and sharding map

Provide event to producer event buffer in order to transmit event to consumer event buffer
**Channel Map 116**

**Stream Type:** Twitter  
**Name:** DeptA  
**NodeID:** U1234567  
**ChannelName:** Twitter_DeptA_U1234567_T0  
**Status:** Active

**Stream Type:** Twitter  
**Name:** DeptA  
**NodeID:** U1234567  
**ChannelName:** Twitter_DeptA_U1234567_T1  
**Status:** Active

**Stream Type:** Twitter  
**Name:** DeptA  
**NodeID:** U555555555  
**ChannelName:** Twitter_DeptA_U555555555_T2  
**Status:** Active

**Stream Type:** Twitter  
**Name:** DeptB  
**NodeID:** U1234567  
**ChannelName:** Twitter_DeptB_U1234567_T0  
**Status:** Active

**Stream Type:** Twitter  
**Name:** DeptB  
**NodeID:** U1234567  
**ChannelName:** Twitter_DeptB_U1234567_T1  
**Status:** Active

**Stream Type:** Twitter  
**Name:** DeptB  
**NodeID:** U555555555  
**ChannelName:** Twitter_DeptB_U555555555_T2  
**Status:** Active

**Stream Type:** Twitter  
**Name:** DeptB  
**NodeID:** U555555555  
**ChannelName:** Twitter_DeptA_U555555555_T3  
**Status:** Active

**FIG. 3A**

---

**Channel 0:** 302a  
**Channel 1:** 302b  
**Channel 2:** 302c  
**Channel 3:** 304a  
**Channel 4:** 304b  
**Channel 5:** 304c  
**Channel 6:** 304d
Sharding Map 118

Stream Type: Twitter
Stream Name: DeptA

Channel-Partitions Map 306

Channel0: Partitions -> 0...32

308a

Channel1: Partitions -> 50-82

308b

Channel2: Partitions -> 33-49, 83-99

308c

Partition To Channel Index 310

0 | 1 | 2 | 3 | 4 | ...... | 30 | 31 | 32

312a

C0

50 | 51 | 52 | 53 | ...... | 80 | 81 | 82

312b

C1

33 | ...... | 49 | 83 | ...... | 99

312c

C2

314d

FIG. 3B
FIG. 4A
Upon adding a new consumer node to the cluster, new consumer node registers with Pending status in consumer map and updates channel map with consumer channels and event streams of interest.

In response to detecting updates to channel map, producer nodes update producer channels based on updated consumer channel configuration in channel map, update sharding map to reflect updated producer channels, and buffer data for transmission over updated producer channels.

In response to detecting updates to sharding map, consumer nodes determine a delta and copy data to be moved into cluster change data map.

In response to detecting updates to cluster change data map, new consumer node copies moved data, clears moved data from cluster change data map and previous consumer node, and changes status to Active in consumer map.

In response to detecting status update in consumer map, producer node begins providing data to producer channel for transmission to corresponding consumer channels on new consumer node.

FIG. 11
## DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document with indication, where appropriate, of relevant passages</th>
<th>Relevant to claim</th>
<th>CLASSIFICATION OF THE APPLICATION (IPC)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>* the whole document *</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>* the whole document *</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The present search report has been drawn up for all claims

<table>
<thead>
<tr>
<th>Place of search</th>
<th>Date of completion of the search</th>
<th>Examiner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Munich</td>
<td>9 December 2015</td>
<td>Anastasov, Yuliyan</td>
</tr>
</tbody>
</table>

**CATEGORY OF CITED DOCUMENTS**

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

- **T**: theory or principle underlying the invention
- **E**: earlier patent document, but published on, or after the filing date
- **D**: document cited in the application
- **L**: document cited for other reasons
- **M**: member of the same patent family, corresponding document
This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on 09-12-2015.

The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

<table>
<thead>
<tr>
<th>Patent document cited in search report</th>
<th>Publication date</th>
<th>Patent family member(s)</th>
<th>Publication date</th>
</tr>
</thead>
<tbody>
<tr>
<td>US 2013047165 A1</td>
<td>21-02-2013</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>US 2013290249 A1</td>
<td>31-10-2013</td>
<td>NONE</td>
<td></td>
</tr>
</tbody>
</table>

For more details about this annex: see Official Journal of the European Patent Office, No. 12/82.
REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

• US 62062515 A [0001]
• US 5826250 A [0038] [0046]
• US 7640222 B [0038] [0046]
• US 8250525 B [0046]