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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
18 October 2018 (18.10.2018)

WIPO | PCT

(10) International Publication Number
WO 2018/188958 A1

(51) International Patent Classification:
H04L 12/26 (2006.01) *G06F 9/54* (2006.01)

(21) International Application Number:
PCT/EP2018/058003

(22) International Filing Date:
28 March 2018 (28.03.2018)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
17166623.3 13 April 2017 (13.04.2017) EP

(71) Applicant: **NOKIA SOLUTIONS AND NETWORKS OY** [FI/FI]; Karaportti 3, 02610 Espoo (FI).

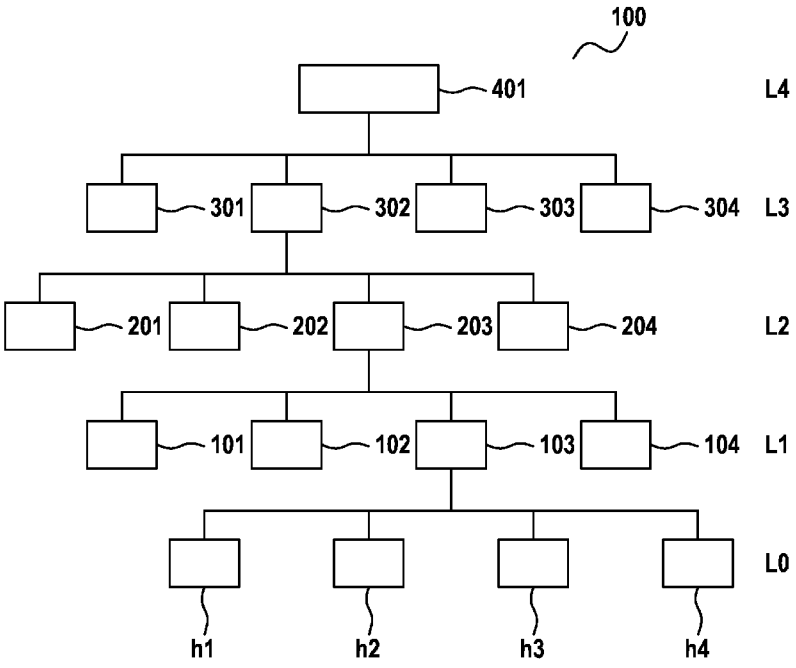
(72) Inventors: **AKKUS, Istemi Ekin**; Nokia Solutions and Networks GmbH & Co. KG, Lorenzstr. 10, 70435 Stuttgart (DE). **RIMAC, Ivica**; Nokia Solutions and Networks GmbH & Co. KG, Lorenzstr. 10, 70435 Stuttgart (DE). **CHEN, Ruichuan**; Nokia Solutions and Networks GmbH & Co. KG, Lorenzstr. 10, 70435 Stuttgart (DE).

(74) Agent: **WETZEL, Emmanuelle**; Nokia Solutions and Networks GmbH & Co. KG, Intellectual Property Business Group, Lorenzstr. 10, 70435 Stuttgart (DE).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA,

(54) Title: A METHOD AND A HOST FOR MANAGING EVENTS IN A NETWORK THAT ADOPTS EVENT-DRIVEN PROGRAMMING FRAMEWORK

Fig. 1



(57) Abstract: The present invention discloses a method of managing events in a network that adopts event-driven programming framework, the network comprising: a plurality of physically distributed hosts, each of which is configured to process at least one event in the network; at least a first entity, configured to maintain a first queue comprising a first plurality of events to be processed by at least one host belonging to a first group of host(s), and at least a second entity, configured to maintain a second queue comprising a second plurality of events to be processed by at least one host belonging to a second group of host(s), wherein, the first group of host(s) comprises the second group of host(s); the method comprising steps of: a) determining a first event in the first queue as an event that is to be processed; b) determining whether the first event satisfies a first condition, the first condition indicating that the first event can be processed by at least a first host belonging to the second group of host(s); b1) if so, retrieving the first event from the first

[Continued on next page]

WO 2018/188958 A1

SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *of inventorship (Rule 4.17(iv))*

Published:

— *with international search report (Art. 21(3))*

queue to the second queue, c) processing the first event on the first host, d) receiving a report of a second event that is newly generated; e) determining whether the second event satisfies a second condition, the second condition indicating that the second event can be processed by at least a second host belonging to the second group of host(s), e1) placing the second event to the second queue and then processing the second event the at the second host, if the second event satisfies the second condition; e2) reporting the second event to the first entity maintaining the first queue, if the second event does not satisfy the second condition.

A method and a host for managing events in a network that adopts event-driven programming framework

Field of the invention

The invention relates to network technology, in particular to a method and a host for managing events in a network that adopts event-driven programming framework.

Background

In a network adopting the serverless computing model, resources are utilized based on demand from the applications. The developers are charged by the amount of computing resources they use (i.e., CPU and memory) during the amount of time their code is running, usually at a second granularity. Adopting the serverless computing model can reduce the costs for developers and enable cloud providers to increase their utilization of resources.

To take advantage of this model, developers usually have to write their applications following an event-driven programming model, where functions of the application are triggered by events happening in the system, such as the upload of an image to server storage or a specific HTTP action by the end user. While this programming model makes developing applications easier by following a “simple-condition corresponding to a simple-action” paradigm.

Some examples of event-driven programming frameworks in the cloud are Amazon Lambda, Google Cloud Functions, Azure Functions and IBM Openwhisk. These systems provide developers with various triggers, including events associated with file storage, HTTP requests, timers and database updates. These systems are designed and promoted for simple scenarios, in which a single trigger can cause one or more short-living functions start executing.

These systems may consider data locality, so that the functions are instantiated at hosts that are close to the hosts storing the data. It is also possible in these systems

to create chains of functions, in which one function invocation creates an event triggering another function and so on.

Summary of the Invention

According to the state of the art, the types of applications that can be developed are limited: applications requiring sequential invocations of several functions can experience increased delays. These delays can be imposed due to the functions triggering each other and sharing/passing application data through a global event queue. An object of the present invention is to enable the developers to reduce the latency between sequential function invocations.

The object of the invention is achieved by the methods and apparatus in the claims.

According to one aspect of the invention, there is provided a method of managing events in a network that adopts event-driven programming framework,
the network comprising: a plurality of physically distributed hosts, each of which is configured to process at least one event in the network, at least a first entity, configured to maintain a first queue comprising a first plurality of events to be processed by at least one host belonging to a first group of host(s), and at least a second entity, configured to maintain a second queue comprising a second plurality of events to be processed by at least one host belonging to a second group of host(s), wherein, the first group of host(s) comprises the second group of host(s); the method comprising steps of: a) determining a first event in the first queue as an event that is to be processed; b) determining whether the first event satisfies a first condition, the first condition indicating that the first event can be processed by at least a first host belonging to the second group of host(s); b1) if so, retrieving the first event from the first queue to the second queue, c) processing the first event on the first host, d) receiving a report of a second event that is newly generated; e) determining whether the second event satisfies a second condition, the second condition indicating that the second event can be processed by at least a second host belonging to the second group of host(s), e1) placing the second event to the second queue and then processing the second event the at the second host, if the second event satisfies the second condition; e2) reporting the second event to the

first entity maintaining the first queue, if the second event does not satisfy the second condition.

In a preferred embodiment, the second group of host comprises only one host, the first host and the second host being the same, and the second queue being a local queue of the host.

In a preferred embodiment, the first condition further includes that a first function that is executed to process the first event is locally stored on the first host, and the second condition further includes that a second function that is executed to process the second event is locally stored on the second host.

The present invention provides a shortcut for events that trigger subsequent event to be processed, such that the delay in processing the subsequent event is reduced. As a result, web applications with low-latency requirements can take advantage of the serverless model (i.e., easier orchestration, management, more cost savings) without paying latency penalties.

In a preferred embodiment, the method further comprises steps of: e3) placing the second event and a status indicator of the second event to the first queue, if the second event satisfies the second condition, wherein, the status indicator indicates to which extent the second group of host(s) has processed the second event, the step a) further comprising steps of: a1) determining to which extent the second group of host(s) has processed the second event based on the status indicator of the second event, a11) preventing the second event in the first queue from being determined as an event that is to be processed, if the status indicator of the second event indicates that the second group of host(s) is still processing the second event or the second group of host(s) has successfully processed the second event; a12) determining the second event in the first queue as an event that is to be processed, if the status indicator of the second event indicates that the second group of host(s) cannot successfully process the second event.

In a preferred embodiment, the status indicator indicates at least one of: 1) whether the first host has started with processing the second event, 2) whether the writing of

a output data has started, 3) whether the first host has finished with processing the second event.

By providing a backup copy of the second event in the higher level queue, the reliability of processing the event is ensured. In case of failure of the second group of host(s), the second event will not be lost and can be processed by another host.

In a preferred embodiment, the step c) further comprises: c1) generating a first output data. The step d) further comprises: d1) receiving the first output data. The method further comprises steps of: e4) storing the first output data on a second data storage that is local to the second group of host(s), if the second event satisfies the second condition; e5) forwarding the first output data to a first data storage that is local to the first group of hosts, if the second event does not satisfy the second condition.

In a preferred embodiment, the step e1) further comprises: generating a second output data after processing the second event the at the second host; the step e3) comprising storing the second output data on the first queue as a status indicator of the second event indicating the processing of the first event has finished.

In a preferred embodiment, the method further comprises a step of: e6) storing the first output data on the first data storage that is local to the first group of hosts, if the second event satisfies the second condition.

Similarly, there is provided a backup copy of the first output data. In case of failure of the first host, the first output data will not be lost and may still be used by another host as input data to process the second event.

In a preferred embodiment, the method further comprises a step of: retrieving the first output data and using the first output data to process the second event.

In a preferred embodiment, the method further comprises steps of: generating metadata from the first output data, writing the generated metadata of the first output data to the first queue as a status indicator of the first event before step e6) to indicate the writing of the first output data is starting.

In a preferred embodiment, if the second group of host(s) has failed in successfully processing the first event, the method further comprises steps of: g) determining whether the metadata of the first output data has been set in the first queue, if so, performing steps h)-j); h) retrieving the first output data from the first data storage and the metadata of the first output data from the first queue; i) determining a metadata based on the first output data retrieved from the first data storage, and j) comparing the metadata determined based on the first output data retrieved from the first data storage and the metadata retrieved from the first queue: j1) if there is a match, setting a status indicator of the first event as the processing of the first event has finished; j2) if there is no match, the method going back to step a), determining the first event in the first queue as an event that shall be processed.

In a preferred embodiment, the method further comprises a step before the step b): determining a workflow of the first event, the workflow indicating a subsequent event that will be generated by processing the first event; the first condition further including that the subsequent event can be processed by at least a second host belonging to the second group of host(s).

The use of hierarchical queues also enables different code/function distribution mechanisms, so that code co-location of frequently interacting functions (i.e., functions in a sequence) is also considered for a variety of web applications (as opposed to today's approach of only considering code/data locality, i.e., code being put in the vicinity of the data it operates on). These mechanisms can focus on optimizing delays in applications as well as resource utilization in cloud infrastructure.

In addition, having multiple levels of queues for messages/events as well as for data enables the cloud operator to separate the infrastructure into distinct clusters, such as different groups of computers being only used for the higher level queue and others being only used for the computation. This separation enables more modularity in maintenance and failure isolation in the cloud infrastructure operating such an event-driven computing framework.

Additionally, the co-location of functions frequently interacting with each other would lead to reuse of the application data that is shared among different functions through the local data layer, further reducing the latency in the processing of the messages/events and ultimately improving application performance.

According to another aspect of the present invention, there is provided a host in a network that adopts event-driven programming framework, wherein, the host subscribes to a first queue and is configured to: a) determine a first event in the first queue as an event that is to be processed; b) determine whether the first event can be processed by the host; b1) if so, the host being further configured to retrieve the first event from the first queue to a local queue of the host; c) process the first event on the host; d) generate a second event; e) determine whether the second event can be processed by the host; e1) if so, the host being further configured to place the second event to the local queue of the host and then process the second event on the host, e2) if not so, the host being further configured to report the second event to an entity maintaining the first queue.

In a preferred embodiment, the host is further configured to: generate a status indicator for the second event and report the second event together with the status indicator to entity maintaining the first queue, if the second event can be processed by the host, the status indicator indicating to which extent the host has processed the second event.

Brief description of the figures

The features and advantages of the invention will be more completely understood by appreciating the following detailed description of preferred embodiments with reference to the figures, wherein

Fig. 1 depicts a schematic block diagram illustrating a network 100 according to an embodiment of the invention;

Fig. 2 depicts a flow chart illustrating a method according to an embodiment of the invention;

Fig. 3 depicts a flow chart of a method according to a preferred embodiment of the present invention;

Fig. 4 depicts a flow chart of the method of recovery from a failover according to the present invention.

Detailed description

Figure 1 shows a schematic block diagram illustrating a network 100 according to an embodiment of the invention.

The network 100 comprises a plurality of physically distributed hosts each of which has a data storage and a processor coupled to the data storage and configured to process at least one event in the network. For simplicity, only four hosts h1, h2, h3 and h4 are shown in Fig 1. A skilled person shall understand, the number of the hosts comprised in the network is not limited to the given example.

In a preferred embodiment, a host may execute the function code that is originally stored on its data storage to process an event in the network. In another embodiment, a host may extract the function code from the data storage of another host, and then execute the extracted function code to process an event in the network. A host may be configured to process more than one event in the network.

The hosts in the network may be categorized into different groups. In one embodiment, the hosts are categorized according to their geographical location. For example, the hosts h1, h2, h3 and h4 may belong to one rack 103. A data center may comprise a plurality of racks. For example, the data center 203 may comprise racks 101, 102, 103 and 104. There may be a plurality of data centers 201, 202, 203 and 204 in a geographical region 302. There may be a plurality of geographical regions 301, 302, 303 and 304 in a higher level geographical region 401.

The criterion that is used to categorize the hosts is not limited to the given example. In another embodiment, the hosts may be categorized according to logical criteria. For simplicity, in the following, the embodiments of the present invention are going to

be described assuming the hosts are categorized according to their geographical location as described above.

Specifically, in the embodiment shown in Fig. 1, each of the hosts maintains a local queue L0 respectively. The local queue L0 each comprises a plurality of events that shall be processed by the respective host. There is an entity in each of the racks maintaining a queue of level 1 L1 respectively. The L1 queue each comprises a plurality of events that shall be processed by the hosts comprised in the respective rack. There is an entity in each of the data centers maintaining a queue of level 2 L2 respectively. The L2 queue each comprises a plurality of events that shall be processed by the hosts comprised in the respective data center. There is an entity in each of the geographical regions maintaining a queue of level 3 L3 respectively. The L3 queue each comprises a plurality of events that shall be processed by the hosts comprised in the respective geographical region. There is an entity in each of the higher level geographical regions maintaining a queue of level 4 L4 respectively. The L4 queue comprises a plurality of events that shall be processed by the hosts comprised in the higher level geographical region 401.

Events are retrieved from a higher level queue to a lower level queue according to the method that will be described in the following, until the event is retrieved to a host and processed on the host. For example, the entity in the geographical region 302 may retrieve a first event from the L4 queue maintained by the entity in higher level geographical region 401 and put the first event in the L3 queue maintained by the entity in the geographical region 302. The entity in the data center 203 may retrieve the first event from the L3 queue maintained by the entity in the geographical region 302 and put the first event in the L2 queue maintained by the entity in the data center 203. The entity in the rack 103 may retrieve the first event from the L2 queue maintained by the entity in the data center 203 and put the first event in the L1 queue maintained by the entity in the rack 103. The first host h1 may retrieve the first event from the L1 queue maintained by the entity in the rack 103 and put the first event in the local queue L0 of the first host h1.

The processing of an event is performed by a host. Different functions are executed to process different events. The functions are preferably stored on the hosts. In one

exemplary embodiment, a first function that is executed to process the first event is locally stored on the first host h1, and the processing of the first event is performed on the first host h1. A second event may be generated by executing the first function on the first host h1. The first host h1 may report the second event to the entity in the rack 103. The entity in the rack 103 may report the second event to the entity in the data center 203. The entity in the data center 203 may report the second event to the entity in the geographical region 302. The entity in the geographical region 302 may report the second event to the entity in the higher level geographical region 401.

Considering a first queue as a lower level queue, a second queue as a higher level queue, it can be generalized that the network 100 further comprises at least a first entity, configured to maintain a first queue comprising a first plurality of events to be processed by at least one host belonging to a first group of host(s), and at least a second entity, configured to maintain a second queue comprising a second plurality of events to be processed by at least one host belonging to a second group of host(s). The first group of host(s) comprises the second group of host(s), i.e. the second group of host(s) is part of the first group of host(s). The entity can be a host or a logical entity. The events in the network can be managed according to the method described in the following with respect to Fig.2.

Fig. 2 depicts a flow chart illustrating a method according to an embodiment of the invention.

In step S2010, a first event in the first queue is determined as an event that is to be processed. The first queue comprises a first plurality of events to be processed by at least one host belonging to a first group of host(s). In one embodiment referring to Fig. 1, the first queue may be L3 queue maintained by the entity in the geographical region 302, and the first group of hosts may be all the hosts belonging to the geographical region 302.

In step S2020, it is determined whether the first event satisfies a first condition, the first condition indicating that the first event can be processed by at least a first host belonging to the second group of host(s). In a preferred embodiment, the first condition further includes that a first function that is executed to process the first

event is locally stored on the first host. If the first event satisfies the first condition, the method proceeds with the step S2030. If the first event does not satisfy the first condition, the method proceeds with other proceedings in step S2021. The other proceedings may be performed to find a host that can process the first event from the first group of host(s), the details of the other proceeding in step S2021 will not be elaborated here for simplicity.

In step S2030, the first event is retrieved from the first queue to the second queue. The second queue comprises a second plurality of events to be processed by at least one host belonging to a second group of host(s).

In an embodiment described referring to Fig. 1, the first queue may be L2 queue maintained by the entity in the data center 203, the second queue may be L1 queue maintained by the rack 103, the first group of hosts may refer to all the hosts belonging to the data center 203, and the second group of hosts may refer to all the hosts belonging to the rack 103 in the data center 203. The first condition for retrieving the first event from the L2 queue maintained by the entity in the data center 203 to the L1 queue maintained by the entity in the rack 103 is that the first event can be processed by at least a first host h1 belonging to the rack 103.

Still referring to Fig.1, the first event may be retrieved from the L4 queue to the L3 queue, from the L3 queue to the L2 queue, from the L2 queue to the L1 queue or from the L1 queue to the L0 queue. A skilled person shall understand, one or more of the above mentioned layers may be omitted. In one embodiment, the first event may be retrieved from L2 queue directly to L0 queue, if the L1 queue is omitted.

Then in step S2040, the first event is processed on the first host h1. A skilled person shall understand that if the second queue is already the local queue L0 of a host, the first event can be directly processed on the host maintaining the local queue L0. If the second queue is L1 queue or higher level queue, steps similar to S2020-S2030 may be performed between step S2030 and step S2040 to retrieve the first event to a lower level queue until the first event arrives at the local queue L0.

In step S2050, a report of a second event that is newly generated is received. A skilled person shall understand that if the second queue is already the local queue L0 of the first host, the second event is generated on the first host, for example in a first function module, then the first function module may report the second event to the entity maintaining the local queue L0 in the first host. If the second queue is L1 queue or higher level queue, the report of the second event is received from the entity maintaining a lower level queue.

In step S2060, it is determined whether the second event satisfies a second condition, the second condition indicating that the second event can be processed by at least a second host belonging to the second group of host(s). In one embodiment, the second condition further includes that a second function that is executed to process the second event is locally stored on the second host.

Still referring to Fig. 1, if the second queue is already the local queue L0 of the first host h1, the first host h1 is the only one host in the second group of host. In that case, the second condition is that the second event can be processed by the first host.

In another embodiment referring to Fig. 1, the second queue is L1 queue maintained by an entity in the rack 103, and a report of the second event is received from the first host h1. In that case, the second condition is that the second event can be processed by the other hosts belongs to the rack 103, i.e. h2, h3 or h4.

If the second event satisfies the second condition, the method proceeds with the step S2070. If the second event does not satisfy the second condition, the method directly proceeds with the step S2100.

In step S2070, the second event is placed to the second queue, and then in step S2080 the second event is processed at the second host.

In an embodiment where the second queue is L1 queue or higher level queue, steps similar to steps S2010-S2030 may be performed between step S2070 and step

S2080 to retrieve the second event to the local queue L0 of the second host that can process the second event.

In an embodiment where the second queue is local queue L0 of the first host h1, the second event is processed on the first host h1 in step S2080.

In step S2090, a status indicator is generated for the second event. The step S2090 may be performed in parallel to the step S2080 or after the step S2080. The status indicator indicates to which extent the second group of host(s) has processed the second event. The status indicator indicates at least one of: 1) whether the first host has started with processing the second event, 2) whether the writing of a output data has started, 3) whether the first host has finished with processing the second event.

In one embodiment of the present invention, a second output data may be generated by processing the second event. The second output data may be written as a status indicator, indicating the processing of the second event has finished. In that case, step S2090 is performed after step S2090.

In step S2100, the second event is reported to the first entity maintaining the first queue. After the first entity receives the second event, the steps S2060-S2100 may be similarly performed on the first entity. If the second event can be processed by at least one host belonging to the first group of host(s), the second event is placed to the first queue. If not, the second event may be further reported to an entity maintaining a higher level queue.

For example, the second event that is generated at the first host h1 may be reported to the entity in the rack 103 if the second event cannot be processed by the first host h1. The entity in the rack 103 may further report the second event to the entity in the data center 203, if the second event cannot be processed by any other host belonging to the rack 103, i.e. the host h2, h3, and h4. The entity in the data center 203 may further report the second event to the entity in the geographical region 302, if the second event cannot be processed by any host belonging to the data center 203.

In an embodiment, if the second event does not satisfy the second condition, the second event is reported to the first entity maintaining the first queue without any status indicator, or in another embodiment, with a status indicator indicating that the second group of host(s) cannot successfully process the second event.

In another embodiment, even if the second event satisfies the second condition, the second event is still reported to the first entity maintaining the first queue together with its status indicator. In this manner, a back up may be provided for the situation of failover. However, if the second event is being processed by the second group of host(s) or the second group of host(s) has successfully processed the second event, the other hosts in the first group of hosts shall not process the second event. Only when the second group of host(s) cannot successfully process the second event, e.g. host fails, the other hosts in the first group of hosts may process the second event.

Fig. 3 shows a flow chart of a method according to a preferred embodiment of the present invention.

According to Fig.3, the status indicator of the second event will be checked to determine whether the second event shall be determined as an event that shall be processed.

In step S2011, the status indicator of the second event will be checked to determine whether the status indicator of the second event indicates that the second group of host(s) is still processing the second event or the second group of host(s) has successfully processed the second event.

If so, the method proceeds with the step S2012. In step S2012, the identification of the event may be updated and the method goes back to step S2011 to check the status indicator of the next event. Detailed proceedings will not be elaborated here. The second event in the first queue is prevented from being determined as an event that is to be processed.

If not so, it suggests that the second group of host(s) cannot successfully process the second event. In that case, the method proceeds with the step S2013. In step S2013,

the second event in the first queue is determined as an event that is to be processed similar to step S2010 as described with respect to Fig. 2.

A skilled person shall understand that after the second event is determined as an event that is to be processed in step S2013, the steps S2020-S2100 described above referring to Fig. 2 may be similarly performed for the second event.

A skilled person shall understand that the second event that is newly generated may also be considered as a normal event in the first queue as the first event. In the following, the embodiments of the present invention will be described with respect to the first event.

In one embodiment of the present invention, a first output data may be generated by processing the first event. The data storage in the network 100 may also be hierarchical, similar as the queues of different levels. The entity maintaining the queue in each level may be further configured to manage the data storage in each level. In another embodiment, the network 100 may comprise another entity to manage the data storage in each level.

The step S2050 may further comprise receiving the first output data. A skilled person shall understand that if the second queue is already the local queue L0 of the first host, the first output data is generated on the first host, for example in a first function module, then the first function module may transmit the first output data to another entity maintaining the data storage of the first host. If the second queue is L1 queue or higher level queue, the first output data is received from the entity maintaining a lower level data storage or the entity maintaining the lower level queue.

In one embodiment, the second event needs to be processed based on the first output data. Therefore, the first output data needs to be retrieved so that the second event can be processed using the first output data. In that case, the first output data shall be stored on or close to the host intended to process the second event.

The step S2070 may further comprise storing the first output data on a second data storage that is local to the second group of host(s). In an embodiment where the

second queue is already the local queue L0 of the first host, the first output data is stored one the data storage of the first host in step S2070. In an embodiment where the second queue is L1 queue, in step S2070, the first output data is stored on the data storage that is local to all the hosts belonging to the rack 103. By doing so, the first output data can be retrieved with less latency.

The step S2100 may further comprise forwarding the first output data to a first data storage that is local to the first group of hosts. In an embodiment, the first output data and the second event that are generated at the first host h1 may be forwarded to the entity in the rack 103 if the second event cannot be processed by the first host h1. The entity in the rack 103 may further forward the first output data and the second event to the entity in the data center 203, if the second event cannot be processed by any other host belonging to the rack 103, i.e. the host h2, h3, and h4. The entity in the data center 203 may further forward the first output data and the second event to the entity in the geographical region 302, if the second event cannot be processed by any host belonging to the data center 203.

In an embodiment, even if the second event satisfies the second condition, the first output data is still stored on the first data storage that is local to the first group of hosts. In this manner, a back up at a higher level data storage may be provided for the situation of failover.

In order to reduce unnecessary or redundant processing of an event during recovery from a failover, in a preferred embodiment of the present invention, any event may be accompanied by a status indicator indicating to which extent the event has been processed. In the following, the embodiments of the present invention will be described with respect to the first event.

In an embodiment, the first output data is generated by processing the first event on the first host h1. Any algorithm, for example, hashing, may be used to generate metadata from the first output data. Then the generated metadata of the first output data will be written to the L1 queue maintained by the entity in the rack 103 as a status indicator of the first event, before storing the first output data on the data

storage that is local to the hosts belonging to the rack 103 to indicate the writing of the first output data is starting.

In the following, one embodiment will be described to elaborate the recovery from a failover according to the present invention.

Assume the first host h1 is now a defunct host. The first event that was retrieved from the L1 queue to the local queue L0 of the first host h1, however the status indicator of the first event in the first queue is not set as successfully processed. Alternatively, the first event may be generated on the first host h1 by processing a foregoing event, the first event is also placed into the L1 queue, and the status indicator of the first event in the first queue is not set as successfully processed.

The failure of the first host will be recognized by the entity in the rack 102 maintaining the L1 queue. As a result, another host (i.e., h3) will be assigned for the events that were previously belonging to the first h1.

The third host h3 is also able to process the first event. After the first host h1 fails, the third host retrieves the first event from the first queue if the status indicator of the first event in the first queue is not set as successfully processed.

Fig. 4 depicts a flow chart of the method of recovery from a failover according to the present invention

In step S4010, it is determined whether the metadata of the first output data has been set in the first queue. If so, the method proceeds with step S4020. Otherwise it means the first host h1 could not set it before it failed. The third host h3 may determine the first event as an event that shall be processed as in step S2010, and then proceeds as described with respect to Fig. 2.

In step S4020, the first output data is retrieved from the data storage that is local to the hosts belonging to the rack 103, and the metadata of the first output data is retrieved from the L1 queue maintained by the entity in the rack 103.

In step S4030, a metadata is determined based on the first output data retrieved from the data storage that is local to the hosts belonging to the rack 103.

In step S4030, it is determined whether the metadata determined based on the first output data retrieved from the first data storage matches to the metadata retrieved from the first queue.

If there is a match, it means the first host h1 could successfully write the output data, but could not set the successful completion flag to indicate the execution has finished. In that case, the third host h3 sets the status indicator of the first event as the processing of the first event has finished in step S4040.

If there is no match, it means the first host h1 could not finish writing of the first output data. Therefore, it is not considered to have fully processed the first event, third host h3 determines the first event in the L1 queue as an event that shall be processed as in step S2010, and then proceeds as described with respect to Fig. 2.

If there is a plurality of events in the local queue of the first host h1 when it fails, the third host may investigate all the events that were in the local queue of h1. Once all old events from the defunct host's partition have been investigated, the third host h3 starts to process the events from the L1 queue as well as from the local queue L0.

The third host h3 were to crash in the middle of this process, then another host will take over the same partition. However, it will retrieve all events after the most recent updates to the status indicator. As a result, the number of events that need to be investigated during the recovery procedure will gradually decrease, even in the presence of multiple host failures.

Following the idea of the present invention, it is also helpful to determine a workflow of the first event, the workflow indicating a subsequent event that will be generated by processing the first event. With the help of the workflow, it is possible to choose a host or a group of hosts close to each other that can process the current event and any other event that may be subsequently generated by processing the current event. Latency caused by the transmission of the event may be further reduced.

In one embodiment, the first condition further includes that the subsequent event can be processed by at least a second host belonging to the second group of host(s), while retrieving the first event from the first queue to the second queue.

Following the idea of the present invention, it is also possible to configure the host or the group of host(s) according to the workflow, so that the events that are frequently occurring in sequence may be processed on the same host or the same group of hosts.

In one embodiment, the second group of host(s) is configured according to the workflow, so that the second group of host(s) can process the first event and the subsequent event(s) that will be generated by processing the first event.

According to another aspect of the present invention, the host h1 in the network 100 subscribes to a first queue L1. There is a local queue at the host h1. The first queue L1 may be a logically-centralized (but physically-distributed) global queue in the system. Host h1 subscribes to the global queue according to the functions it has locally available. The global queue is segmented into shards/partitions. Each subscribing host gets assigned one or more partitions in a mutually exclusive way (i.e., no two hosts get the same partitions).The host h1 is configured to:
a) determine a first event in the first queue L1 as an event that is to be processed;
b) determine whether the first event can be processed by the host h1;
b1) if so, the host h1 retrieves the first event from the first queue L1 to a local queue L0 of the host h1;
c) process the first event on the host h1;
d) generate a second event;
e) determine whether the second event can be processed by the host h1;
e1) if so, the host h1 places the second event to the local queue of the host h1 and then process the second event on the host h1,
e2) if not so, the host h1 reports the second event to an entity maintaining the first queue L1.

The host h1 may also generate a status indicator for the second event and report the second event together with the status indicator to the entity maintaining the first queue L1, if the second event can be processed by the host h1, the status indicator indicates to which extent the host h1 has processed the second event.

5

Claims

1. A method of managing events in a network that adopts event-driven programming framework,
the network comprising:
a plurality of physically distributed hosts, each of which is configured to process at least one event in the network,
at least a first entity, configured to maintain a first queue comprising a first plurality of events to be processed by at least one host belonging to a first group of host(s), and at least a second entity, configured to maintain a second queue comprising a second plurality of events to be processed by at least one host belonging to a second group of host(s), wherein, the first group of host(s) comprises the second group of host(s);
the method comprising steps of:
a) determining a first event in the first queue as an event that is to be processed;
b) determining whether the first event satisfies a first condition, the first condition indicating that the first event can be processed by at least a first host belonging to the second group of host(s);
b1) if so, retrieving the first event from the first queue to the second queue,
c) processing the first event on the first host,
d) receiving a report of a second event that is newly generated;
e) determining whether the second event satisfies a second condition, the second condition indicating that the second event can be processed by at least a second host belonging to the second group of host(s),
e1) placing the second event to the second queue and then processing the second event the at the second host, if the second event satisfies the second condition;
e2) reporting the second event to the first entity maintaining the first queue, if the second event does not satisfy the second condition.

2. A method according to claim 1, wherein, the second group of host comprises only one host, the first host and the second host being the same, and the second queue being a local queue of the host.

3. A method according to claim 1, wherein, the first condition further includes that a first function that is executed to process the first event is locally stored on the first

host, and the second condition further includes that a second function that is executed to process the second event is locally stored on the second host.

4. A method according to claim 1, further comprising steps of:
e3) placing the second event and a status indicator of the second event to the first queue, if the second event satisfies the second condition,
wherein, the status indicator indicates to which extent the second group of host(s) has processed the second event,
the step a) further comprising steps of:
a1) determining to which extent the second group of host(s) has processed the second event based on the status indicator of the second event,
a11) preventing the second event in the first queue from being determined as an event that is to be processed, if the status indicator of the second event indicates that the second group of host(s) is still processing the second event or the second group of host(s) has successfully processed the second event;
a12) determining the second event in the first queue as an event that is to be processed, if the status indicator of the second event indicates that the second group of host(s) cannot successfully process the second event.

5. A method according to claim 4, wherein the status indicator indicates at least one of:
1) whether the first host has started with processing the second event,
2) whether the writing of a output data has started,
3) whether the first host has finished with processing the second event.

6. A method according to claim 1, the step c) further comprising:
c1) generating a first output data,
the step d) further comprising:
d1) receiving the first output data,
the method further comprising steps of:
e4) storing the first output data on a second data storage that is local to the second group of host(s), if the second event satisfies the second condition;
e5) forwarding the first output data to a first data storage that is local to the first group of hosts, if the second event does not satisfy the second condition.

7. A method according to claim 4, the step e1) further comprising:
- generating a second output data after processing the second event the at the second host;
the step e3) comprising storing the second output data on the first queue as a status indicator of the second event indicating the processing of the first event has finished.

8. A method according to claim 6, further comprising a step of:
e6) storing the first output data on the first data storage that is local to the first group of hosts, if the second event satisfies the second condition.

9. A method according to claim 6, further comprising a step of:
- retrieving the first output data and using the first output data to process the second event.

10. A method according to claim 8, further comprising steps of:
- generating metadata from the first output data,
- writing the generated metadata of the first output data to the first queue as a status indicator of the first event before step e6) to indicate the writing of the first output data is starting.

11. A method according to claim 10, if the second group of host(s) has failed in successfully processing the first event, the method further comprising steps of:
g) determining whether the metadata of the first output data has been set in the first queue, if so, performing steps h)-j);
h) retrieving the first output data from the first data storage and the metadata of the first output data from the first queue;
i) determining a metadata based on the first output data retrieved from the first data storage, and
j) comparing the metadata determined based on the first output data retrieved from the first data storage and the metadata retrieved from the first queue:
j1) if there is a match, setting a status indicator of the first event as the processing of the first event has finished;

j2) if there is no match, the method going back to step a), determining the first event in the first queue as an event that shall be processed.

12. A method according to any one of claims 1-10, further comprising a step before the step b):
- determining a workflow of the first event, the workflow indicating a subsequent event that will be generated by processing the first event;
the first condition further including that the subsequent event can be processed by at least a second host belonging to the second group of host(s).

13. A method according to any one of claims 1-10, further comprising steps of:
- determining a workflow of the first event, the workflow indicating a subsequent event that will be generated by processing the first event;
- configuring the second group of host(s) according to the workflow, so that the second group of host(s) can process the first event and the subsequent event that will be generated by processing the first event.

14. A host in a network that adopts event-driven programming framework, wherein, the host subscribes to a first queue and is configured to:
a) determine a first event in the first queue as an event that is to be processed;
b) determine whether the first event can be processed by the host;
b1) if so, the host being further configured to retrieve the first event from the first queue to a local queue of the host;
c) process the first event on the host;
d) generate a second event;
e) determine whether the second event can be processed by the host;
e1) if so, the host being further configured to place the second event to the local queue of the host and then process the second event on the host,
e2) if not so, the host being further configured to report the second event to an entity maintaining the first queue.

15. A host according to claim 14, further configured to:

generate a status indicator for the second event and report the second event together with the status indicator to entity maintaining the first queue, if the second event can be processed by the host,

the status indicator indicating to which extent the host has processed the second event.

Fig. 1

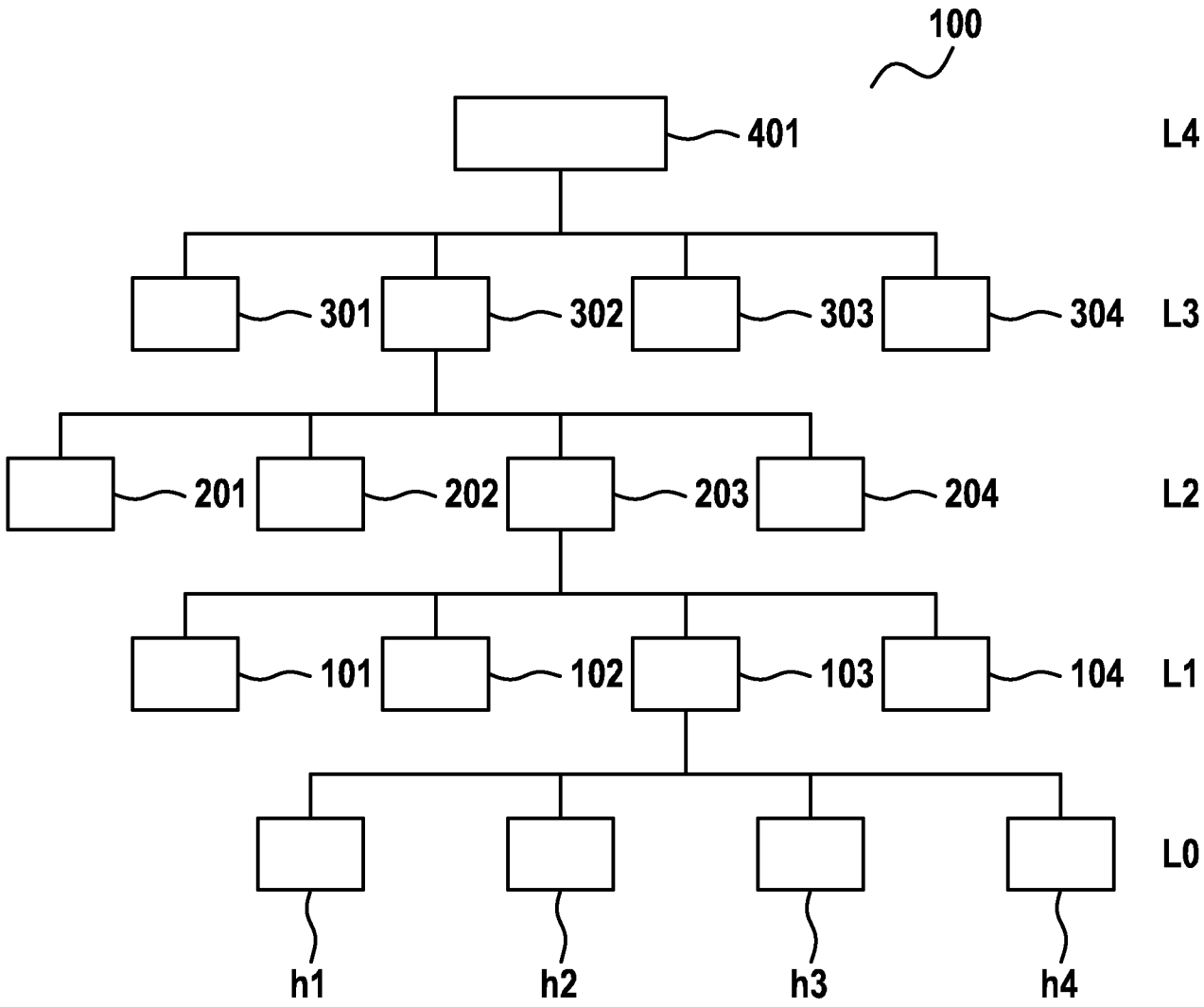

100
401
L4
301
302
303
304
L3
201
202
203
204
L2
101
102
103
104
L1
L0
h1
h2
h3
h4

Fig. 2

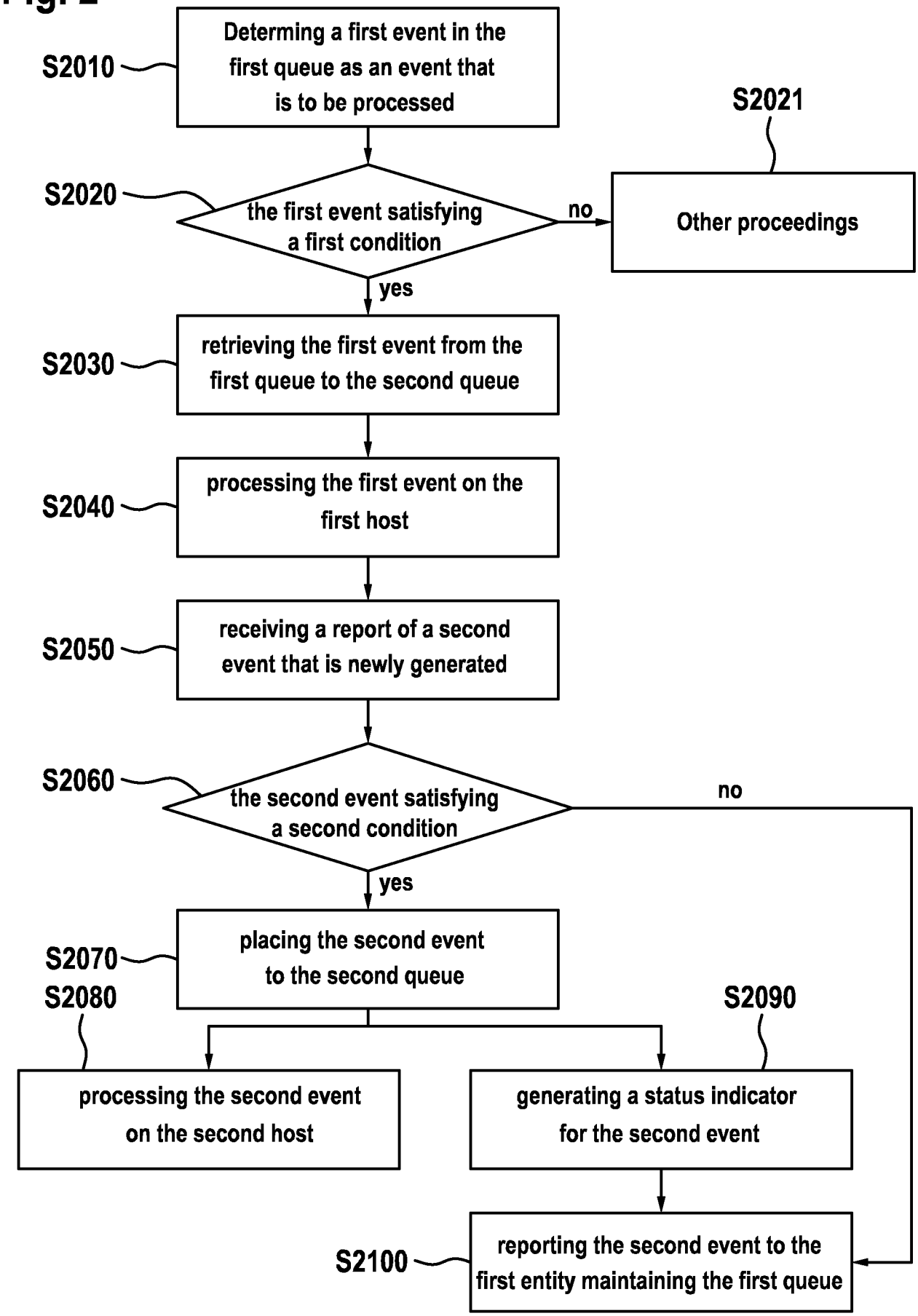

S2010
Determing a first event in the first queue as an event that is to be processed
S2020
the first event satisfying a first condition
no
S2021
Other proceedings
yes
S2030
retrieving the first event from the first queue to the second queue
S2040
processing the first event on the first host
S2050
receiving a report of a second event that is newly generated
S2060
the second event satisfying a second condition
no
yes
S2070
placing the second event to the second queue
S2080
S2090
processing the second event on the second host
generating a status indicator for the second event
S2100
reporting the second event to the first entity maintaining the first queue

Fig. 3

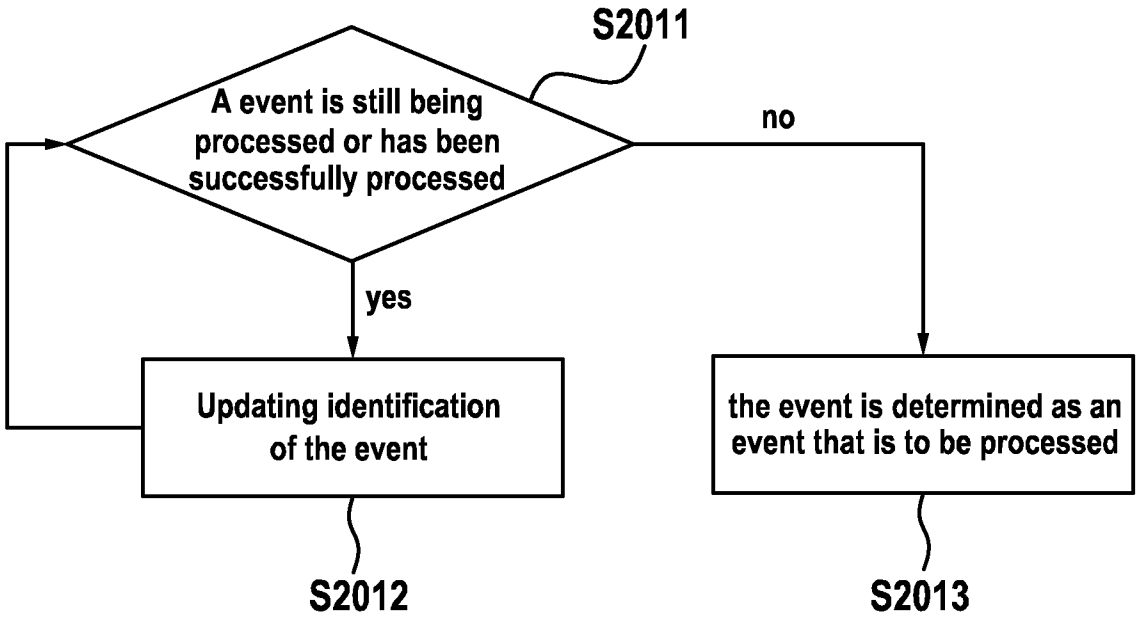
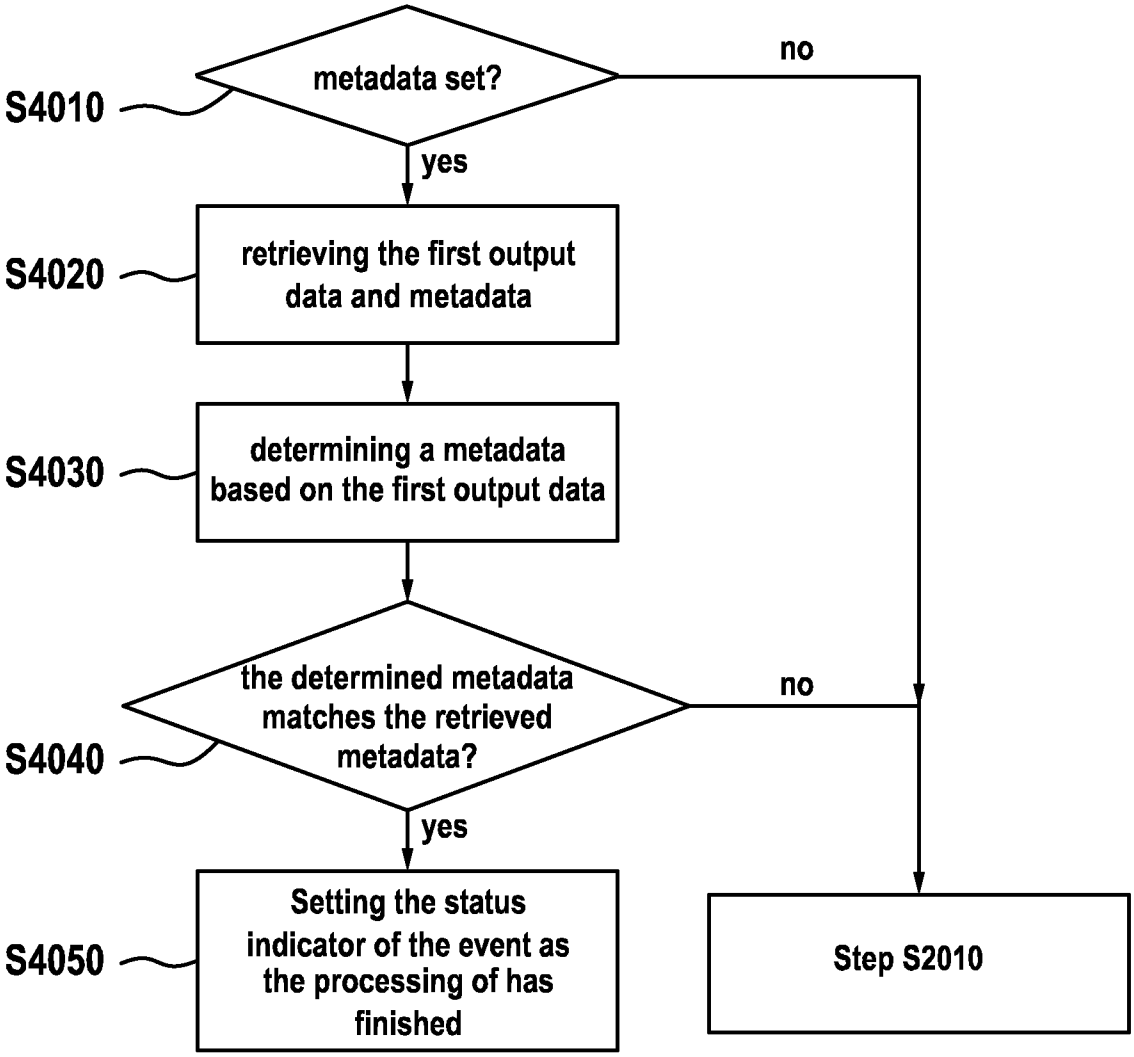

S2011
A event is still being processed or has been successfully processed
no
yes
Updating identification of the event
the event is determined as an event that is to be processed
S2012
S2013

Fig. 4


S4010
metadata set?
no
yes
S4020
retrieving the first output data and metadata
S4030
determining a metadata based on the first output data
S4040
the determined metadata matches the retrieved metadata?
no
yes
S4050
Setting the status indicator of the event as the processing of has finished
Step S2010

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2018/058003

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04L12/26 G06F9/54
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H04L G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	STEPHEN MARZ ET AL: "Reducing Power Consumption and Latency in Mobile Devices Using an Event Stream Model", ACM TRANSACTIONS ON EMBEDDED COMPUTING SYSTEMS, ACM, NEW YORK, NY, US, vol. 16, no. 1, 13 October 2016 (2016-10-13), pages 1-24, XP058298751, ISSN: 1539-9087, DOI: 10.1145/2964203	1-3,6,8, 9,14
A	Introduction, Chapters 2-5, Fig.2-5	4,5,7, 10-13,15

-/--

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
23 May 2018	05/06/2018

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Plata-Andres, Isabel

2

Form PCT/ISA/210 (second sheet) (April 2005)

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CHRISTIAN POELLABAUER ET AL: "Flexible cross-domain event delivery for quality-managed multimedia applications", ACM TRANSACTIONS ON MULTIMEDIA COMPUTING COMMUNICATIONS ANDAPPLICATIONS, ASSOCIATION FOR COMPUTING MACHINERY, US, vol. 1, no. 3, 1 August 2005 (2005-08-01), pages 248-268, XP058193812, ISSN: 1551-6857, DOI: 10.1145/1083314.1083316 Chapter 3. -----	1-15
A	RODRIGO N CALHEIROS ET AL: "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", 20110101, vol. 41, no. 1, 1 January 2011 (2011-01-01), pages 23-50, XP007917900, DOI: 10.1002/SPE.995 Chapter 3.4 -----	1-15