

A COMMUNICATIONS SERVER, A METHOD, A USER DEVICE, AND SYSTEM

Technical Field

The invention relates generally to the field of communications. One aspect of the invention relates to a communications server apparatus for data aggregation.

Another aspect of the invention relates to a method, performed in a communications server apparatus for data aggregation. Another aspect of the invention relates to a communications device for data aggregation. Another aspect of the invention relates to a payment gateway. Another aspect of the invention relates to a payment system. Another aspect of the invention relates to a method of data aggregation.

Background

Various forms of data aggregation exist.

For example, in US10929418, a method of aggregating data for electronic transactions is disclosed. US7315849 US10515367 and US20210117990 also disclose methods of processing data. One technical problem that may exist in the art is how to compute aggregated quantities for time series data without having to do extensive database queries for each transaction.

Summary

Embodiments may be implemented as set out in the independent claims. Some optional features are defined in the dependent claims.

Implementation of the techniques disclosed herein may provide significant technical advantages. Advantages of one or more aspects may include:

- faster calculation of metrics for big data;
- allows much faster user fraud detection in financial transactions;

allows user to user comparison in financial transactions for the same or similar vendor without an increase in transaction latency; and/or

allows adaptive promotions to be customised to each user without an increase in transaction latency.

In at least some implementations, the techniques disclosed herein may allow for:

- the technical solution of less hardware required for the technical problem of calculating metrics for big data in real time;
- the technical solution of less latency for the technical problem of calculating metrics for big data in real time;
- the technical solution of faster execution of risk score for the technical problem of detecting fraud in financial transactions;
- the technical solution of improved accuracy of an offer or encouragement to a user, based on the technical problem of how to determine user preferences from a timing pattern of transactions;
- the technical solution of improved accuracy of an offer or encouragement based on the technical problem of how to determine user preferences from comparison of time series transactions for other users at the same or similar locations;
- the technical solution of offering a promotion to cleaner / green vendor based on the technical problem of identifying customer who prefer green vendors; and/or
- the technical solution of reducing the data traffic transmitted between servers, or between applications or modules within a server, based on the technical problem of calculating metrics for big data in real time;
- the technical solution of reducing the latency of data traffic transmitted between the server and the in-vehicle units based on the technical problem of faster calculating metrics for big data;
- the technical solution of reduced data centre energy requirements based on the technical problem of optimally calculating metrics for big data.

In an exemplary implementation, the functionality of the techniques disclosed herein may be implemented in software running on a server communications apparatus, such as a cloud-based database. The software which implements the functionality of the techniques disclosed herein may be contained in a computer program, or computer program product – which operates the database instances on each server node in the cloud. When running on, for example, a cloud server, the hardware features of the server may be used to implement the functionality described below, such as using the server network communications interface components to establish the secure communications channel for aggregating metrics in relation to a large amount of transaction data stored in the database.

In one aspect there is provided a communications server apparatus for data aggregation, the communications server comprising a processor and a memory, the communications server apparatus being configured, under control of the processor, to execute instructions stored in the memory, to:

- store a plurality of timing-based parameters associated with a time series of events, separately for each user;
- update the parameters in response to a new event;
- calculate one or more metrics using the updated parameters, and
- determine if the metrics satisfy at least one criterion.

In another aspect there is provided a method performed in a communications server apparatus for data aggregation, the method comprising, under control of a processor of the communications server apparatus:

- storing a plurality of timing based parameters associated with a time series of events, separately for each user;
- updating the parameters in response to a new event;
- calculating one or more metrics using the updated parameters; and
- determining if the metrics satisfy at least one criterion.

In a further aspect there is provided a user communications device for communicating with a payment gateway comprising a processor and a memory, the communications device being configured, under control of the processor, to execute instructions stored in the memory to:

- request the payment gateway to process a transaction;

- if a plurality of transactions for that user, based on a plurality of timing based parameters associated with a time series of events, stored separately for that user, indicate an acceptable risk verdict, receive a transaction confirmation from the payment gateway; and

- if the plurality of transactions indicate an unacceptable risk verdict, receive a transaction declined from the payment gateway.

In a still further aspect there is provided a system for online credit card transaction fraud prevention comprising:

- a database configured to store a plurality of parameters associated with a separate data store of a time series of transactions;

- a payment gateway configured to receive a request for a payment on a new transaction, and to update the parameters with each new transaction;

- a risk score server configured to determine a risk score for the transaction based on the parameters, wherein the determination includes calculating an average time between transactions for a user associated with the new transaction based on the parameters; and

- wherein the payment gateway is configured to approve or decline the new transaction based on the risk score.

In a still further aspect there is provided a payment system, comprising:

- a communications server;

- at least one driver communications device;

- at least one user communications device; and

a communications network equipment configured for the communications server, the at least one driver communications device, and the at least one user communications device, to establish communication with each other therethrough;

wherein the driver communications device comprises a first processor and a first memory, the driver communications device being configured, under control of the first processor, to execute first instructions stored in the first memory to:

send data regarding a completed trip to the communications server;

and wherein the communications server comprises a second processor and a second memory, the communications server being configured, under control of the second processor, to execute second instructions stored in the second memory to:

initiate a new transaction based on the completed trip data,

update a plurality of stored parameters associated with a separate data store of a time series of historical transactions, based on data associated with the new transaction,

determine a risk score for the transaction based on the parameters, and one or more risk rules, and

approve or decline the new transaction based on the risk score;

and wherein the user communications device comprises a third processor and a third memory, the user communications device being configured, under control of the third processor, to execute third instructions stored in the third memory to:

receive data from the communications server confirming the transaction has been approved or declined.

In a still further aspect there is provided a method, performed in a payment gateway, the method comprising, under control of processors for one or more server instances:

storing a plurality of parameters associated with a separate data store of a time series of historical transactions;

receiving a request for a new transaction;

updating the plurality of parameters, based on data associated with the new transaction;

determining a risk score for the transaction based on the parameters, and one or more risk rules; and

approving or declining the new transaction based on the risk score.

Brief Description of the Drawings

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

Fig. 1 is a schematic block diagram illustrating an exemplary transportation service.

Fig. 2 is a schematic block diagram illustrating an exemplary communications server for the transportation service.

Fig. 3 is a schematic block diagram illustrating an exemplary PSP router architecture.

Fig. 4a and 4b are flow diagrams of how first transaction time and last transaction time bin map key will be updated.

Fig. 5 is a flow diagram of how minimum and maximum transaction time bin map key will be updated.

Fig. 6 is a sequence diagram of how an event takes place in case the transaction data is unordered with respect to time.

Fig. 7 is a sequence diagram of how an event takes place in case the transaction data is ordered with respect to time.

Fig. 8 is a sequence diagram of how data is read in real time flow diagram illustrating an exemplary processing flow.

Detailed Description

The techniques described herein are described primarily with reference to use in taxi, ride hailing, ride sharing, food delivery, online groceries, service / trade exchanges, and pet transport, but it will be appreciated that these techniques have a broader reach and can be usefully implemented in other fields where event data

aggregation (such as transaction data aggregation) may be required. Generally, this might be the case in any e-commerce site with a significant number of transactions.

Figure 1 shows an exemplary architecture of a transportation service system 100, with a number of users each having a communications device 104, a number of drivers each having a user interface communications device 106, a server 102 (or geographically distributed servers) and communication links 108 connecting each of the components. Each user contacts the server 102 using a user software application (app) on the communications device 104. The user app may allow the user to enter their pick-up location, a destination address, one or more service parameters, and/or after-ride information such as a rating. The one or more service parameters may include the number of seats of the vehicle, the style of vehicle, level of environmental impact and/or what kind of transport service is desired. The user app may also be used to order delivery of food or other items. Each driver contacts the server 102 using a driver app on the communications device 106. The driver app allows the driver to indicate their availability to take jobs, information about their vehicle, their location, and/or after-ride info such as a rating. The server 102 may then match users to drivers, based on, for example: geographic location of users and drivers, maximising revenue, user or driver feedback ratings, weather, driving conditions, traffic levels / accidents, relative demand, environmental impact, and/or supply levels. This allows an efficient allocation of resources because the available fleet of drivers is optimised for the users' demand in each geographic zone.

Referring to Figure 2, further details of the components in the system of Figure 1 are now described. The communications apparatus 100 comprises the communications server 102, and it may include the user communications device 104 and the driver communications device 106. These devices are connected in the communications network 108 (for example, the Internet) through respective communications links 110, 112, 114 implementing, for example, internet communications protocols. The communications devices 104, 106 may be able to communicate through other

communications networks, including mobile cellular communications networks, private data networks, fibre optic connections, laser communication, microwave communication, satellite communication, etc., but these are omitted from Figure 2 for the sake of clarity.

The communications server apparatus 102 may be a single server as illustrated schematically in Figure 2. Alternatively, the functionality performed by the server apparatus 102 may be distributed across multiple physically or logically separate server components. In the example shown in Figure 2, the communications server apparatus 102 may comprise a number of individual components including, but not limited to, one or more microprocessors 116, a memory 118 (e.g. a volatile memory such as a RAM, and/or longer term storage such as SSD (Solid State or Hard disk drives (HDD)) for the loading of executable instructions 120, the executable instructions defining the functionality the server apparatus 102 carries out under control of the microprocessor 116. The communications server apparatus 102 also comprises an input/output module 122 allowing the server to communicate over the communications network 108. User interface 124 is provided for user control and may comprise, for example, computing peripheral devices such as display monitors, computer keyboards and the like. The server apparatus 102 also comprises a database 126, the purpose of which will become readily apparent from the following discussion.

The user communications device 104 may comprise a number of individual components including, but not limited to, one or more microprocessors 128, a memory 130 (e.g., a volatile memory such as a RAM) for the loading of executable instructions 132, the executable instructions defining the functionality the user communications device 104 carries out under control of the microprocessor 128. The user communications device 104 also comprises an input/output module 134 allowing the user communications device 104 to communicate over the communications network 108. A user interface 136 is provided for user control. If

the user communications device 104 is, say, a smart phone or tablet device, the user interface 136 will have a touch panel display as is prevalent in many smart phone and other handheld devices. Alternatively, if the user communications device 104 is, say, a desktop or laptop computer, the user interface 136 may have, for example, computing peripheral devices such as display monitors, computer keyboards and the like.

The driver communications device 106 may be, for example, a smart phone or tablet device with the same or a similar hardware architecture to that of the user communications device 104. Alternatively, the functionality may be integrated into a bespoke device such as a taxi fleet management terminal.

Thus, it will be appreciated that Figures 1 and 2 and the foregoing description illustrate and describe a communications server apparatus 102 comprising a microprocessor 116 and a memory 118, the communications server apparatus 102 being configured, under control of the microprocessor 116, to execute instructions 120 stored in the memory 118, to:

- store a plurality of timing-based parameters associated with a time series of events, separately for each user;
- update the parameters in response to a new event;
- calculate one or more metrics using the updated parameters; and
- determine if the metrics satisfy at least one criterion.

Further, it will be appreciated that Figures 3 to 5 illustrate and describe a method performed in a communications server apparatus 102, the method comprising, under control of a microprocessor 116 of the server apparatus 102:

- storing a plurality of timing-based parameters associated with a time series of events, separately for each user;
- updating the parameters in response to a new event;
- calculating one or more metrics using the updated parameters; and

determining if the metrics satisfy at least one criterion.

Embodiments enable more efficient computation of user-specific parameters where a large number of user-specific events occur over time. This may improve the in-app experience for the user by reducing delays in serving content to the user or by increasing responsiveness to user input. For example, in an e-commerce environment with high numbers of transactions, even a slight improvement in user experience (such as card acceptance rate, transaction response latency, service uptime) makes a big difference in the everyday life of customers.

In terms of fraud detection in such financial transactions, it is necessary to check through a very large data set of past transactions to determine a risk verdict or anomaly score. Unfortunately, with the time it takes to execute queries across the whole data set, real-time fraud detection can be difficult. Thus, there may be a dilemma to choose between making fraud detection as robust as possible, against transaction latency and/or customer satisfaction.

Thus, one example embodiment may use aggregate timing-based parameters associated with a time series of events, separately for each user. Each variable may be aggregated for a predetermined time period, e.g.: for each day or each hour.

In this way, if it is desired to determine the average time between transactions for a user over a specific time period (e.g. the past 7 days), this can easily be determined without having to do a full data set query. This may include one or more advantages including:

- faster calculation of metrics for big data;
- allows much faster user fraud detection in financial transactions;
- allows user to user comparison in financial transactions for the same or similar vendor without an increase in transaction latency; and/or

allows adaptive promotions to be customised to each user without an increase in transaction latency.

To give context, several possible problems or use cases are listed below:

1. With opt-in by the user, understanding the pattern of the user activity by considering the average, minimum and maximum time gap between the user's activities, which can help to serve more relevant content (e.g. custom offers and products) to the user, with the user's consent.
2. With user permission, analysing the behaviour of the user (e.g. when and how the user interacts with other entities in the system, such as merchants or restaurants) to improve service delivery and/or user experience.
3. Detecting a bot or other malicious entity masquerading as a legitimate user by checking the average time gap between the user's activities (e.g. transactions).

Analysing user activity for bespoke offers

In one example, the user may opt in for custom offers and rewards based on their activity. For example, user events such as switching between a user app (designed based on one or more embodiments described herein) and some other app can be logged. These may be analysed in accordance with one or more embodiments described herein, and if it is observed from the analysis that the average time between the events (going and coming back to the user app) is less in a given time interval compared to another time interval (e.g. the last x days or the last x weeks), this may indicate that the user is comparing a product/price with other vendors (which may offer products or services within the other app or apps) and is planning to buy. In that case the user can be presented (within the user app) with a custom discount or cashback so as to encourage the user to commit to the transaction on the user app.

By being able to fetch data metrics in real time, this allows a determination of which offer, if any, needs to be presented to the user with little or no additional latency.

Likewise, the frequency of use of an app by the user may be measured by checking the minimum time gap between events when the user is on the app. If the time gap is in days, then the user has not used or transacted on the app for more than a day, and an email or in-app notification with an offer or surprise may be sent to attract the user. Another option is to compare the current week's data with the previous week's, or multiple past weeks', data.

For example, suppose that the past 2 months of metadata (aggregated timing-based parameters) are saved and the current date is 18 Aug 2021. Multiple past weeks (9-15 Aug, 2-8 Aug etc) of metadata - e.g. minimum, maximum, and average time difference between events - are retrieved and compared with the corresponding values for the current week (16-18 Aug). As the data can be retrieved in real time (that is within milliseconds), which allows decision making about what information to present to the user. For example, the 95th percentile latency for fetching the above information may, in an example system, be about 15ms.

Comparing user activity with other users

If the user is going to interact with an entity by doing a transaction, we can determine the avg/min/max data points of the user previously for that entity to determine the behaviour. For example, if user X is interacting with restaurant A, then we can compare the current behaviour (e.g. as reflected by the timing profile of the interactions) with the past behaviour of X with A. In another example, the behaviour of X with A can be compared with the behaviour of another user (say Y, Z, etc) with A to help understand the pattern of the market and also to help entities (such as merchants) to be more effective.

Detecting malicious activity:

Suppose that a malicious user came to know about a rate limiting threshold associated with fraud detection in financial transactions. That malicious user might try to restrict

the transaction within the threshold limit and repeat (e.g. through automation) the activity after the time that the threshold refreshes.

This behaviour might be implemented in a bot for example. In such a case the transaction frequency will typically be periodic such that the average time between transactions should be the same across multiple days. So, the behaviour for a given user can be monitored, and if a malicious pattern exists then the user can be restricted or reported.

For example, if User A does 5 transactions on 2 Aug then the average transaction gap for those 5 transactions can be computed, and compared with the average transaction gap on previous days (say 1 Aug, 31 July, 28 July etc). In the case of automation, the transaction will typically follow the same time gap across intervals, and this can be easily detected.

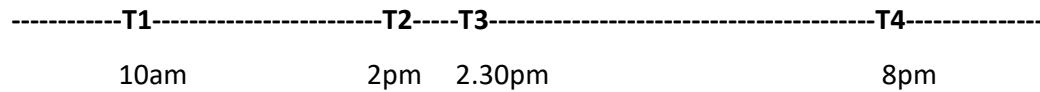
The timing information of various users can be fed into a ML model, and this can then be used to further identify patterns that are different from general trends (e.g. for the entire population of users, or for specific subgroups of users). Also, some scenarios where some patterns are not reflective of normal human behaviour can be detected. For example, if a min gap of, say, a few milliseconds is detected between (say) the 2 top users it might be flagged as suspicious activity; as a human would still take a few seconds to do the same activity.

The model can then also be trained based on the transaction timings of fraudulent users and then be used to detect anomalies in future transactions.

Time scale variables

One or more embodiments may aggregate timing-based parameters associated with a time series of events, separately for each user. For example, this may include fetching 'minimum/maximum time gap' based queries in constant time for a given time window, and storing these variables separately for each user.

To understand this, suppose that events are generated for a user - in this example, consequent on the performance of transactions - in the following order during a given day:



The time difference between transactions will be as follows:

$$T2-T1 = 4 \text{ hours}$$

$$T3-T1 = 0.5 \text{ hour}$$

$$T4-T3 = 5.5 \text{ hours}$$

For the above data, the minimum time gap will be 0.5 hours, and the maximum time gap will be 5.5 hours for that day. The data for each day can be stored temporarily, in a rotating FIFO fashion. For example, depending on the requirements of the application, the system may store the data for up to 7 days, 30 days, 60 days, 3 months, 6 months, or 12 months.

The prior art techniques of querying the data set for every transaction will inherently increase the transaction latency as the dataset increases. These queries may alternatively be dependent on aggregate functions, such as max or min, processing all the data after it has been collected. The time window for a given query may be days, weeks or even months. Hence as the time window increases, the lookup data size will also increase, and so will the latency. Prior art solutions may not be able to determine a minimum/maximum time gap between transactions without querying the entire historical transaction data set.

Below are sample queries, generated according to prior art techniques, on a time series database (Timescale DB) which is optimized (*breaks down the table into*

multiple chunks) for handling time series queries but which takes significant time for queries. The query time increases as the time interval increases.

Note: Columns included in the Queries are indexed.

Time Interval	2 days
Query	<pre>SELECT COALESCE(max(time), DATE '0001-01-01') as maxTimeGap FROM (SELECT time, time - LAG(time) OVER(ORDER BY time ROWS BETWEEN 1 PRECEDING AND CURRENT ROW) as diff FROM dummy_table WHERE from_user_id = 'fromUser450' AND currency = 'IDR' AND status = 'success' AND txn_mode ='QR' AND time >= '2021-07-15 00:00:00' AND time <= '2021-07-16 23:59:59')t;</pre>
Query time	3518.362 ms (00:03.518)
Data size	1935473 records

Time Interval	7 days
Query	<pre>SELECT COALESCE(max(time), DATE '0001-01-01') as maxTimeGap FROM (SELECT time, time - LAG(time) OVER(ORDER BY time ROWS BETWEEN 1 PRECEDING AND CURRENT ROW) as diff FROM dummy_table WHERE from_user_id = 'fromUser450' AND currency = 'IDR'</pre>

	<i>AND status = 'success' AND txn_mode = 'QR' AND time >= '2021-07-15 00:00:00' AND time <= '2021-07-21 23:59:59')t;</i>
Query time	<i>13957.305 ms (00:13.957)</i>
Data size	<i>6774182 records</i>

Time Interval	<i>30 days</i>
Query	<i>SELECT COALESCE(max(time), DATE '0001-01-01') as maxTimeGap FROM (SELECT time, time - LAG(time) OVER(ORDER BY time ROWS BETWEEN 1 PRECEDING AND CURRENT ROW) as diff FROM dummy_table WHERE from_user_id = 'fromUser450' AND currency = 'IDR' AND status = 'success' AND txn_mode = 'QR' AND time >= '2021-07-01 00:00:00' AND time <= '2021-07-31 23:59:59')t;</i>
Query time	<i>50896.195 ms (00:50.896)</i>
Data size	<i>28802029 records</i>

As can be seen from the above, whether it is 4s, 14s or 51s, the latency introduced by such queries would be unacceptable in a real-time user interaction (such as a financial transaction).

Fraud detection detailed example

An example architecture of a risk system 300 for fraud detection in financial transactions is shown in Figure 3, together with data flows between certain system components. The risk system 300 is responsible for fraud detection and prevention. The risk system 300 may store past data in a NoSQL database, such as an Aerospike database 308. The database 308 is used to maintain aggregated timing-based parameters (also referred to herein as time metadata) for different users. In some embodiments, database 308 is a flash-optimized and in-memory open source distributed key value NoSQL database.

In some embodiments, the time metadata may be updated on a per-transaction (or other discrete event) basis, for the particular time window (e.g. day) during which the transaction (or other event) takes place. That is, for each new event that occurs, the time metadata for the time window are retrieved and, if necessary, updated based on the timing data for the new event. Examples of processes for performing such updates will be described in further detail below with reference to Figures 4 to 8.

In some embodiments, database 308 may be used for temporary storage of data, while an additional database (not shown in Figure 3) may be used for persistent storage of the data. For example, a Timescale database may be used for persistent storage. The Timescale database (built on top of Postgres) , specialised for time series data, is used for lookup in the background for a small-time range when required.

When a user initiates a transaction, for example using a digital wallet 302 or credit card in an online financial transaction, the digital wallet 302 requests authorisation for the transaction from a merchant website. The merchant website will in turn forward the request to the risk system 300 according to one or more embodiments.

In that case the risk API 310 will in real-time read the current datapoints from the database 308. The risk API 310 in real-time will pass the data points to a machine

learning (ML) or data science (DS) module 312. The ML/DS module 312 may be part of, and/or operated by the same entity as, risk system 300. Alternatively, the ML/DS module 312 may be an external module to which data are transmitted for analysis. The datapoints may be used to train one or more ML models, for rules evaluation, and for real-time fraud detections. The one or more ML models may ingest various data points to detect fraud. For example, data such as average time between transactions, minimum time between transactions, and/or maximum time between transactions for a particular time window (e.g. 14 days) may be used.

Based on different values of avg, min, and/or max transaction time, the ML model may give an anomaly score. If the score is above a threshold, then the events and/or time period being scored can be categorised as fraud. For example, if the min time is too small over a period of time, then it can be categorized as some sort of bot. Similarly with avg time, if the current avg time of a user transaction differs from multiple past values of avg time then it can be assumed that the user account has been compromised and an unwanted transaction is taking place. Likewise other conditions with respect to time can be enforced and subsequently a score can be determined.

The ML model may give a high anomaly score whenever suspicion is detected with respect to historical data, which can be used to prevent the transaction from taking place.

The data science module 312 will in real-time return an anomaly score for that transaction. Real time in this context may mean, in an example system, a 95th percentile latency of about 50ms to apply the ML model. The risk API 310 will apply a risk rule to the anomaly score to return a risk verdict to the merchant website. The result of the processing will then be forwarded in real-time to the user's digital wallet 302. For example, if the risk verdict is such that the transaction is blocked, an indication of this will be forwarded to the user's digital wallet 302 and displayed on the user's device.

As all these data/aggregates are derived and evaluated in real time for each of the user transactions, latency is a crucial factor which needs to be kept in mind as it will affect overall user transaction latency.

In parallel to the fraud detection, with each transaction the datapoints in the database 308 are updated. First the details of the transaction (including time scale data and a user ID) are pushed into an event streaming platform 304 (such as kafka), and are then processed by a risk worker module 306 into the various metrics, which are then processed and the database 308 updated as necessary. Figures 4 to 8 give examples of how the risk worker module 306 processes the transaction data to update the metadata in the database 308.

The risk-worker module 306 does post-processing of transactions. Once the transaction is complete then the corresponding transaction record with details is pushed to an event processing system, the output of which is then consumed by the risk worker application 306 and processed to update the various data points in database 308 as necessary and also inserted in persistent storage that are needed for risk evaluation for future transactions.

In some applications it may be useful to aggregate the minimum/maximum time gap, in constant time without actually loading or scanning through the whole dataset for any time window.

This may be generalised to dividing a problem or objective into subproblems and maintaining the “metadata for each sub problem”. When it is determined that a given time window is of interest, the system may combine the stored metadata of each sub problem so as to cover the given time window, to allow a desired analysis to be done in respect of the metadata for the given time window.

In the following, the below terminology is used:

- Set: Maintain the key/entities for which metadata will be stored. i.e., “user” or can be composite of multiple entities such as “user.merchant”
- Bin: Level at which metadata will be stored, e.g. day level. A person skilled in the art can determine whatever bin size is reasonable and best suited for each application. Embodiments are not restricted to day level.
- Bin Map Key: field/data points inside the bin that will contain/hold the values. e.g. *first-txn-time*, *min-txn-time-diff*
- Ordered Data: Data in which the sequence of events is the same as the time order in which they occurred.
- Un-Ordered Data: Data in which the sequence of events is not the same as the time order in which they occurred. For example, a transaction may have an earlier timestamp than another transaction, but may be received in the event stream at a later time due to network transmission delays or dropped packets.

In some applications, the sub problem or “bin” is day level. The metadata or “Bin Map Key” may be:

1. First transaction time of the day *alias first-txn-time: this field will contain the time at which the first transaction took place on a day.*
2. Last transaction time of the day *alias last-txn-time: this field will contain the time at which the last transaction took place on a day.*
3. Minimum transaction time duration of the day *alias min-txn-time-diff: this field will contain the minimum time gap among all the transactions which took place on a day.*
4. Maximum transaction time duration of the day *alias max-txn-time-diff: this field will contain the maximum time gap among all the transactions which took place on a day.*

All the above-mentioned fields may be stored in a map structure for all the time windows (e.g. days) in which the user is transacting, for each user.

Suppose that a minimum/maximum time gap between all the transactions done by a user over a week is required. To obtain this, the stored last seven “day level metadata” may be retrieved (from database 308, or from a persistent-storage database) and combined to derive the minimum/maximum time gap for the week. Reading of the aggregates for minimum/maximum time gap can be done from database 308 in a single call using the primary key, hence offering the least latency, as shown below:

User1:

120321 --> day level	130321	140321
{ first-txn-time: “time in epoch”, last-txn-time: “time in epoch”, min-txn-time-diff: “time diff in nano sec”, max-txn-time-diff: “time diff in nano sec” }	{ first-txn-time: “time in epoch”, last-txn-time: “time in epoch”, min-txn-time-diff: “time diff in nano sec”, max-txn-time-diff: “time diff in nano sec” }	{ first-txn-time: “time in epoch”, last-txn-time: “time in epoch”, min-txn-time-diff: “time diff in nano sec”, max-txn-time-diff: “time diff in nano sec” }

User2:

120321 (date at day level)	130321	140321
{ first-txn-time: “time in epoch”, last-txn-time: “time in epoch”, min-txn-time-diff: “time diff in nano sec”, max-txn-time-diff: “time diff in nano sec” }	{ first-txn-time: “time in epoch”, last-txn-time: “time in epoch”, min-txn-time-diff: “time diff in nano sec”, max-txn-time-diff: “time diff in nano sec” }	{ first-txn-time: “time in epoch”, last-txn-time: “time in epoch”, min-txn-time-diff: “time diff in nano sec”, max-txn-time-diff: “time diff in nano sec” }

For the above structure, the above-defined parameters will be:

- Set: User1, User2 (tables)
- Bins: 120321, 140321 (columns). 120321 is the date format for: 12 March 2021
- Bin Map Key: *first-txn-time, min-txn-time-diff*

There are two scenarios when dealing with time series data as shown in Figure 5:

1. Ordered Data 502
2. Unordered Data 504

In ordered data, the sequence is maintained so only the last transaction details saved need to be considered. Based on this, the current transaction details can be processed as shown in Figure 7. In accordance with Figure 7, the Bin field will be updated in aerospike 308 as follows:

- *first-txn-time: min(first-txn-time, current transaction time)*
- *last-txn-time: max(last-txn-time, current transaction time)*
- *min-txn-time-diff: min(min-txn-time-diff, current transaction time - last-txn-time)*
- *max-txn-time-diff: max(min-txn-time-diff, current transaction time - last-txn-time)*

In unordered data the sequence is not maintained, such that the events (e.g. transactions) are received in random, or otherwise non-sequential, order. As the time sequence is uncertain, sorted data is required in order to calculate the time gap. So, timescale database 602 is used for retrieving the minimum/maximum data for a small part (day duration) as shown in Figure 6. This is done using the processes shown in figures 4a for first txn time 402, and Figure 4b for last txn time 404.

In accordance with Figure 6, the Bin field will be updated as follows:

- *first-txn-time: min(first-txn-time, current transaction time)*

- *last-txn-time*: $\max(\text{last-txn-time}, \text{current transaction time})$
- *min-txn-time-diff*: $\min(\text{min-txn-time-diff}, \text{min time gap data retrieved from timescale 602})$
- *max-txn-time-diff*: $\max(\text{max time gap data retrieved from timescale 602}, \text{current transaction time})$

All these metadata fields within a bin will be updated in a single call to database 308. When all the above metadata are available in database 308, the following is one approach for calculating minimum/maximum time gap 802 over a time interval as shown in Figure 8.

For calculating the minimum time gap between transactions, the following metadata can be used:

1. First transaction time of the day (i.e., *first-txn-time*)
2. Last transaction time of the day (i.e., *last-txn-time*)
3. Minimum transaction time duration of the day (i.e., *min-txn-time-diff*)

First transaction time and last transaction time are required to calculate the duration across different days of the window.

Suppose that a minimum transaction time gap from 1 Aug to 4 Aug for a user is desired. Consider that the following metadata (of day level bins) are available in database 308:

Date	1 Aug	2 Aug	3 Aug	4 Aug
first-txn-time	F1	F2	F3	F4
last-txn-time	L1	L2	L3	L4
min-txn-time-diff	M1	M2	M3	M4

Minimum time gap will be = $\text{Minimum}(M1, M2, M3, M4, (F2-L1), (F3-L2), (F4-L3))$. Here $F2-L1$ gives the time gap between transaction L1 which was the last transaction of 1 Aug and F2 which was the first transaction of 2 Aug. The minimum time gap for any given rolling window which spans across a long duration (like weeks or months) can be calculated in like fashion.

Calculating the Maximum time gap between transactions in a given time window will be the same as calculating the min. The only difference is instead of min operator, max operator will be used.

Suppose that a maximum time gap from 1 Aug to 4 Aug is desired. Consider that the following metadata are available:

Date	1 Aug	2 Aug	3 Aug	4 Aug
first-txn-time	F1	F2	F3	F4
last-txn-time	L1	L2	L3	L4
max-txn-time-diff	M1	M2	M3	M4

Maximum time gap will be = $\text{Maximum}(M1, M2, M3, M4, (F2-L1), (F3-L2), (F4-L3))$. Here $F2-L1$ gives the time gap between transaction L1 which was the last transaction of 1 Aug and F2 which was the first transaction of 2 Aug. The maximum time gap for any given rolling window which spans across a long duration, like weeks or months, can be calculated in like fashion.

The above metadata approach can be used to calculate the Average time gap between the user events (e.g. transactions) taking place over a given time interval. For calculating average time gap between transactions, the following metadata can be used:

1. First transaction time of the day (i.e., *first-txn-time*)

2. Count of transactions taking place in that day

For calculating the average time gap below will be the formula:

Avg time gap = (Current txn time - first txn time of the window) / No of transactions in that window.

Suppose that an Average time gap from 2 Aug to 4 Aug is desired. Consider that the following metadata are available:

Date	1 Aug	2 Aug	3 Aug	4 Aug
first-txn-time	F1	F2	F3	F4
count-of-txn	C1	C2	C3	C4

first-txn-time of the window = F2 (as the window starts from 2nd Aug)

No of transactions in that window = (C2+C3+C4)

Avg time gap= (Current txn time - F2) / (C2+C3+C4)

The average time gap for any given rolling window which spans across a long duration, like weeks or months, can be calculated in like fashion.

As will be appreciated by a person skilled in the art, a range of datapoints or metadata may be aggregated according to the requirements of the application. In at least the use applications mentioned above, there are one or more technical solutions provided including aggregated datapoints or metadata associated with a time series of events, that are at least temporarily stored, separately for each of a plurality of users.

It will be appreciated that the invention has been described by way of example only. Various modifications may be made to the techniques described herein without departing from the spirit and scope of the appended claims. The disclosed

techniques comprise techniques which may be provided in a stand-alone manner, or in combination with one another. Therefore, features described with respect to one technique may also be presented in combination with another technique.

Claims

1. A payment system, comprising:
 - a communications server;
 - at least one driver communications device;
 - at least one user communications device; and
 - a communications network equipment configured for the communications server, the at least one driver communications device, and the at least one user communications device, to establish communication with each other therethrough;wherein the driver communications device comprises a first processor and a first memory, the driver communications device being configured, under control of the first processor, to execute first instructions stored in the first memory to:
 - send data regarding a completed trip to the communications server;and wherein the communications server comprises a second processor and a second memory, the communications server being configured, under control of the second processor, to execute second instructions stored in the second memory to:
 - initiate a new transaction based on the completed trip data,
 - update a plurality of stored parameters associated with a separate data store of a time series of historical transactions, based on data associated with the new transaction,
 - determine a risk score for the transaction based on the parameters, and one or more risk rules, and
 - approve or decline the new transaction based on the risk score;and wherein the user communications device comprises a third processor and a third memory, the user communications device being configured, under control of the third processor, to execute third instructions stored in the third memory to:
 - receive data from the communications server confirming the transaction has been approved or declined.

2. The payment system of claim 1 further configured to:
store the plurality parameters, separately for each user;
calculate one or more metrics using the updated parameters, and
determine if the metrics satisfy at least one criterion for the risk score.
3. The payment system of claim 2 wherein the parameters include a minimum time between transactions and a maximum time between financial transactions, over a predetermined period.
4. The payment system of claim 3 wherein the at least one criterion includes a determination as to whether each financial transaction is fraudulent or not, based on the minimum time between financial transactions and maximum time between financial transactions.
5. The payment system of any preceding claim wherein the metrics include an average time between financial transactions over the selected period.
6. The payment system of claim 5 wherein the parameters include, over a selected time period: a first financial transaction time, a last financial transaction time, and a count of financial transactions.
7. The payment system of claim 6 wherein the average time between financial transactions is calculated based on the difference between the first financial transaction time and the last financial transaction time divided by the number of financial transactions.
8. The payment system of claim 1 wherein the transactions are online user usage patterns, and the at least one criterion includes a determination as to whether the user has become less loyal and/or is currently contemplating a transaction.

9. The payment system of claim 8 further comprising displaying a user specific promotion if the user is determined to have become less loyal and/or is currently contemplating a transaction.

10. The payment system of claim 1 wherein the transaction are user usage patterns, and the at least one criterion includes a comparison of a plurality of users' transaction behaviour in relation to one or more selected credit card merchants.

11. A method, performed in a payment gateway, the method comprising, under control of processors for one or more server instances:

- storing a plurality of parameters associated with a separate data store of a time series of historical transactions;

- receiving a request for a new transaction;

- updating the plurality of parameters, based on data associated with the new transaction;

- determining a risk score for the transaction based on the parameters, and one or more risk rules; and

- approving or declining the new transaction based on the risk score.

12. The method of claim 11, wherein the plurality parameters are stored separately for each user, the method further comprising

- calculating one or more metrics using the updated parameters; and

- determining if the metrics satisfy at least one criterion for the risk score.

13. A computer program or computer program product comprising instructions for implementing the method of claim 11.

14. A non-transitory storage medium, storing instructions, which when executed by a processor, causes the processor to perform the method of claim 11.

15. A user communications device for communicating with a payment gateway comprising a processor and a memory, the communications device being configured, under control of the processor, to execute instructions stored in the memory to:

request the payment gateway to process a transaction;

if a plurality of transactions for that user, based on a plurality of timing based parameters associated with a time series of events, stored separately for that user, indicate an acceptable risk verdict, receive a transaction confirmation from the payment gateway; and

if the plurality of transactions indicate an unacceptable risk verdict, receive a transaction declined from the payment gateway.

16. A system for online credit card transaction fraud prevention comprising:

a database configured to store a plurality of parameters associated with a separate data store of a time series of transactions;

a payment gateway configured to receive a request for a payment on a new transaction, and to update the parameters with each new transaction;

a risk score server configured to determine a risk score for the transaction based on the parameters, wherein the determination includes calculating an average time between transactions for a user associated with the new transaction based on the parameters; and

wherein the payment gateway is configured to approve or decline the new transaction based on the risk score.

17. The system of claim 16 wherein the determination includes comparing the average time between transactions for the user across a plurality of selected time periods, to identify patterns or indications of automation or fraud.