N-diff for streaming trade logs

Reconcile trade log entries arriving in independent streams.

Input

Streams are assumed to be iterators that can be reset to a specific point in time. There may be an arbitrary number of them and they should be non-blocking.

To simplify debugging, each stream origin will be referred to with a unique identifier **O_ID**.

Record Format

Each record **O_R** will consist of at least the following fields:

- **TS** - Timestamp (YYYY-MM-DD HH:MM:SS.sssssss) in UTC
- **T_ID** - Trade ID, unique to each trading transaction
- **RF_*** - One or more fields that will be compared for reconciliation.
- **DF_*** - One or more fields that will be included in output for debugging purposes.

Inter-stream Jitter

There may be an inter-stream jitter of up to **J** seconds for TS in records that are otherwise identical. Any time skew within this window should be considered a non-difference.

An outside jitter window of **E** seconds (E > J) will be used to limit memory usage and raise alarms if an input stream is blocked.

Output

The system will note any differences between streams as a record **D_R** in an output stream. This output stream will include the **O_ID** and **O_R** for each stream. Each output record will include a list of inconsistent fields **DF**. If an input record is missing, the inconsistent field set will include "_missing".

For cases where the inter-stream jitter is greater than **E**, the output stream will include two or more **D_Rs** each with one or more “_missing” DFs and null O_Rs.

Example

T_ID, [DF, …], [O_ID.0, O_R.0], [O_ID.1, O_R.1], …, [O_ID.N, O_R.N]

Logic Overview

Data Structures

- **nextRecords** - A list of the next log record from each log stream [TS, O_ID, O_R], sorted by timestamp.
- **pendingTrades** - A dict of mappings from unreconciled trade IDs to a (sometimes partial) list of associated log records [T_ID -> [O_R.0, O_R.1, …, O_R.N]] from each log stream.
- **pendingRecords** - A list of each unreconciled log record [TS, O_ID, T_ID] previously read from each log stream in the past E seconds, sorted by timestamp.

Reconciliation

1. Compare the TS and RF_* fields in each of the log records for an individual trade.
    a. Text RF_* fields will use a case-insensitive comparison.
    b. Numeric RF_* fields will be transformed with abs() before comparison.
    c. If any do not match, emit D_R records.
2. Remove the related entries from pendingTrades and pendingRecords.

Main Loop

1. Reconcile any pendingTrades that have pendingRecords with a timestamp that is older than E
    a. Use the next TS from nextRecords as a basis for pendingRecords ages.
2. Attempt to populate nextRecords with a new record from any log stream that isn't currently represented in it.
3. Fetch the next record from nextRecords
    a. Attempt to backfill nextRecords from whichever log stream it originated from.
    b. Add it to pendingRecords
    c. If the trade is not in pendingTrades, add it and then add the record to it.
    d. Otherwise, add the record to the existing trade in pendingTrades.
    e. If the trade in pendingTrades has records from all log streams, reconcile it.

Reliability Considerations

State Persistence

State required for restarting this service should be persisted externally on a regular basis (~1 minute of data?). This state should include the most recently read timestamp from each log stream.

Restarting

When restarting, each log stream should be rewound to the stored timestamps mentioned above, minus E seconds. This may result in some duplicate reconciliation output but should ensure that no reconciliation is missed.

Instrumentation

Metrics should be emitted into an external timeseries store on a regular basis (~1 wallclock minute or ~10000 log records?). These metrics should include:

- Process runtime
- Resource consumption (cpu, memory, …)
- The count of “good” and “not good” reconciled trades.
- The count of each DF value.
- A histogram of timestamp jitter (max_ts - min_ts) for “good” and “not good” reconciled trades.
- The count of records read from each log stream.
- The count of missing records from each log stream.
- A histogram of record read latencies for each log stream.
- A histogram of (walltime - timestamp) for each log stream.
- A histogram of the latency of writes to the external state persistence store.
- A histogram of the latency of a single main loop iteration.

Monitoring/Alerting

The primary mechanism for operational monitoring will be checking that metrics are emitted as frequently as expected, and that the good/not-good counts are not too small/large. Alerts should also be defined for the latencies of log stream reads and main loop iteration.

Open Questions

- Should the case-insensitive/abs() filters for RF_* fields be configurable?
- How frequently should metrics be emitted?