1. Requirement to calculate unique users per minute, day, week, month, year.

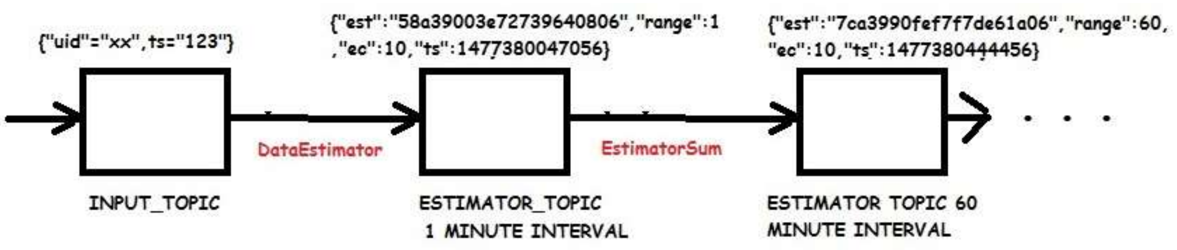
Estimator is apropriate bitmap structure (hash table) available in HyperLogLog and Linear counting algorithms. Idea is to produce estimator every minute (minimum time interval).

Record structure produced directly (online) from Kafka topic (stream reader) can be:

{"ts":<timestamp>, "range":<range>,"ec":<ecvalue>,"est":<estimator>}

Value <timestamp> unix time.
Value <range> is in minutes (1, 5, 60, 1440, ...) or coded (1 for 1 min, 2 for 5 min, 3 for 60 min etc).
Value <ecvalue> is parameter of HyperLogLog (<=30) or Linear counting (>30).
Value <estimator> is hex dump of serialized object (bitmap).

Output is written to separate resulting Kafka topic every minute.


{"uid"="xx",ts="123"}
{"est":"58a39003e72739640806","range":1 ,"ec":10,"ts":1477380047056}
{"est":"7ca3990fef7f7de61a06","range":60, "ec":10,"ts":1477380444456}
DataEstimator
EstimatorSum
INPUT_TOPIC
ESTIMATOR_TOPIC 1 MINUTE INTERVAL
ESTIMATOR TOPIC 60 MINUTE INTERVAL

To sum estimators, "ec" parametar (bitmap number of bytes in the case of Linear counting) should be the same.

Estimator addition is scalable. Cardinality estimation is based on resulting estimator for given time range. Resulting estimator has the same size as any estimator used in calculation.

Proposed granularity (configurable):

- 1 minute interval – last 2 days-+ (60*24*2 = 2880 records)
- 5 minute interval – current week (12*24*7 = 2016 records)
- 60 minute interval – last 3 months (max 24*31*3 = 2231 records)
- 1 day interval – last 6 years (max 366*6 = 2196 records)

Every category is written into dedicated Kafka topic. One criteria is to maintain similar number of records in every topic (in this case ~ 2200 records).

Separate batch processing (cron job) maintains granularity of estimators (addition and purge). Optional backup on permanent media is possible.

2. There can be several estimator records containing same timestamp and range.

Query from application reads all records that match given criteria and performs estimator addition to produce resulting cardinality estimation.

To minimize error, resulting cardinality should be less than expected cardinality (provided as input parametar)

Any batch processing (late packets, packet fixes, restore from backup, etc) produces same way separate estimator records with appropriate time stamps and range according to configured granularity. Another batch job can consolidate estimators performing addition and purge.

Scalability is achieved same way:

Separate Kafka nodes read from one or more Kafka topic and produce estimator records in proposed manner.

3. Expected cardinality

Expected cardinality is the most important parameter of HyperLogLog and Linear counting algorithms.

http://highscalability.com/blog/2012/4/5/big-data-counting-how-to-count-a-billion-distinct-objects-us.html

HyperLogLog - an improved version of LogLog that is capable of estimating the cardinality of a set with accuracy = 1.04/sqrt(m) where expected cardinality is **m = 2^b**. So we can control accuracy vs space usage by increasing or decreasing **b** <= 30.

Linear counting – parameter n is size of bit array in bytes. Expected cardinality is **m = 8*n** So we can control accuracy vs space usage by increasing or decreasing **n**.

4. Other issues

In current version of DataEstimator.java conversion of bit array to string and vice versa is done locally (for example 1011100011100 to 171c).

Providing not sufficient number of bytes for bit array causes integer overflow.

To handle large bit array, we need to use Kafka gzip of the topic and serialize / deserialize using Kafka methods.