A Method for Fast Index Lookup in Databases through Linear Approximation Window

Krassimir Tzvetanov[1], Michael Momchedjikov[2]

Abstract – **in this paper we present a fast and simple algorithm for fast timestamp-to-sequence-number matching based on the linear window approximation.**

Keywords – **database, search, lookup, approximation**

I. Introduction

Have you ever had a database that gets some 15×10^6 records per week? Did you ever try to make a simple search in it? The following method presents a very efficient way of improving performance by locating the indexed sequential number of the record(s) your looking for.

In large databases it is necessary to index data so that you queries can run faster. Unfortunately many times this is not possible if the data you are trying to index is very diverse and not correlating. In the traffic analysis system presented in [1], there were collected about 1.5-2 million records per day. Every record keeps an indexed sequential number, a time/date field and other not essential to the algorithm data. The time/data field expresses the count of seconds that have passed since the epoch in seconds since Jan 1 1970 seconds [2]. Keeping in mind that a day has 86400 and the database collects 1.5 million records per second it is not practical to index by that field. The first big problem we encounter is the size of the index itself. It's easy to calculate the for only one week we will have 604 800 groups of indexes and each one of them will hold average of 18 different records. 18 out of millions is negligible number it's not worth indexing. Over a half a million groups in an index is something not easy to search through (consider multiplying that by the number of weeks you plan to keep accounting information for). The time for searching through the index file is considerable. There is even no way to fit the index into the memory so it has to, always, be searched from the disk.

The experiment showed that on MySQL[3] database the index created on the timestamp field takes 16-18% of the database itself.

II. Presenting the Method

In this paper we show a method that allows significantly faster lookup of the value that must be indexed. It helps you find the sequence number of the record in the database based on the timestamp of the record. It's obvious that the sequence number multiplied by the size of the record gives you the exact location of the record.

$$O = I_r * R_s \ , \tag{1}$$

where O is the offset in the file, I_r is the sequence of the record and R_s is the size of the record.

If the traffic accounting records were inserted into the database with a constant speed, it would also be possible to use that formula. Unfortunately this is not possible since traffic has very different characteristics. On Figure 1 the number of records received on an average work and weekend days of the university network is shown. As it can be seen during work days the distribution is far from linear.

(It is also interesting to note that if you have data from very long time period (over 10 days) the daily oscillations will be invisible in the first few steps, even though they will become a cause of errors in the next steps. In my further research we are considering dynamically changing the distribution law used for approximation.)

In this case linear[3] approximation will yield very incorrect results which will increase search time, etc.

At the same time the distribution is linear for an infinitely short time frame. Therefore the smaller the time frame is – the closer to linear the distribution will be.

You can also consider that if you constantly decrease the time frame (i.e. search window) you'll be getting better results.

This is what exactly the algorithm does. It makes its first linear approximation with a window the size of the whole database. As expected, the result will be far from precise. However, this information is sufficient so that we can narrow the search window for the next step. The new window is calculated on the basis of the approximated value. If the new window doesn't contain the value we are looking for, then we rerun this step using a window with borders: the old value of the border and the new value (the one that is not correct). This window approximately has the same size of the one, which we would have used if there had been no error.

The approximation is done in the following manner: first, we find the average records per hour (2):

$$R_r = (R_t - W_{ti})/(W_{tx} - W_{ti}) \ , \tag{2}$$

where R_r is the record rate (record per second), W_{ti} and W_{tx}

[1]Krassimir Tzvetanov holds Bs from the Faculty of Communictions and Communications Technologies, Technical University, Sofia; Email: krassi@tu-sofia.bg

[2]Michael Momchedjikov is with the Faculty of Communictions and Communications Technologies, Technical University, Kliment Ohridski 8, 1000 Sofia, Bulgaria; Email: mom@tu-sofia.bg

[3]If linear approximation is replaced by approximation following the distribution law of the variable we use. The results should be much better. This is a focus of future work.

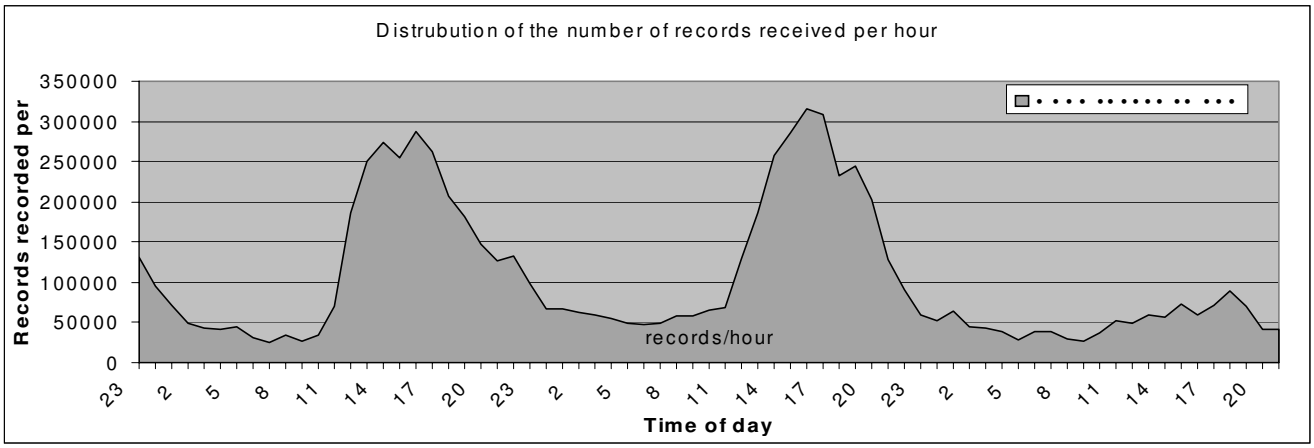

Fig. 1. Distribution of the number of records received per hour

are the lower and upper time borders of the window. R_t is the time/date of the record we are looking for.

The next step is to approximate the sequence number (I_a)(3) based on the this rate:

$$I_a = W_{ii} + (W_{ix} - W_{ii}) * R_r \ , \qquad (3)$$

where W_{ii} and W_{ix} are the sequence numbers of the lower and upper borders.

As a next step, we calculate the new window borders (4)(5). Note that it is not symmetric around the approx. value.

$$B_{ln+1} = \max(I_a - (I_a - W_{ii}) * 0.15, B_{ln}) \qquad (4)$$

$$B_{un+1} = \min(I_a + (W_{ix} - I_a) * 0.20, B_{un}) \qquad (5)$$

Where B_l and B_u denote the lower and upper border. Where n refers to the current step and $n+1$ to the next step of the algorithm.

We repeat this procedure recursively, making smaller and smaller windows – therefore achieving better approximation. There is a minimum distance below which is better to continue searching consequently without approximation. This value is based on the speed of the CPU, disk array, and network activity. It is difficult to calculate this value precisely. However the experiment shows that this value varies between 200 and 500 records.

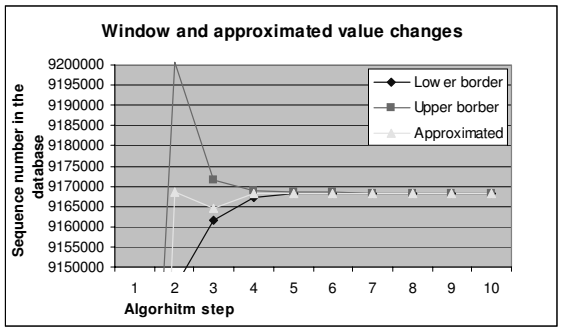

Fig. 2. Window and approximated value changes

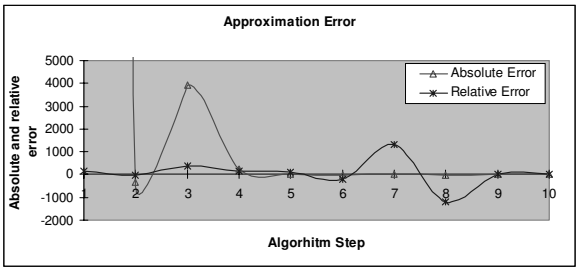

Fig. 3. Approximation Error

III. Results from an Experiment Conducted in TU-Sofia

Below are included graphs based on data taken from the university network during 5 consecutive days and a test run of the program. (Both workdays and weekends are included to make distribution more complex; the data is about 12 million records).

On Figure 2 you can see how both the approximated value and the window change. Both reach close proximity really fast.

On Figure 3 you can see the absolute and relative error during each step of the algorithm. The line, marked with $\triangle$, represents the absolute error in prediction. It goes down exponentially. (The value after the first step, not shown on the graph, is 236 128 records absolute error).

Note that the minimum distance value of the algorithm is achieved at step 6 (or 7). This means that the algorithm is more time consuming than regular sequential search from this moment on.

The line, marked with $*$, represents the relative error (absolute error/how wide the search window is). As you can see on the first step the relative error is small because we have data from several days which hides the daily variations.

When we continue after the 6th step, the algorithm starts to work inefficiently. This is because the rules for calculating the window are not symmetric. This helps while working with large data arrays but makes the algorithm unstable

when working with small amounts of data. This is not really an issue since there is one more reason that justifies why we should stop before that point. Even more, those quotients can be changed.

On the test system the proposed algorithm finds the record about 35-40 times faster than the one without an index (the test database includes only some 10 million records). If you have more records in the database – you'll get better results.

Because the method of searching through the database without an index is sequential, the time increases for the newer records linearly because they occupy the last positions in the database. The search time of proposed method does not depend on the location of the record in the database. It depends solely on the number of records in the database but not in a linear way. The search time grows very slowly related to the increase (of records) in number.

IV. Further Research

As already noted in our future work we plan to find ways to improve the approximation method. It is interesting to replace the linear approximation with approximation with normal distribution. It is also interesting research ways to dynamically change the approximation method.

Acknowledgements

We would like to express our special thanks to the Ministry of Education and Science for their continuing support.

References

[1] Krassimir Tzvetanov, 2003, System for Traffic Analysis and Accounting in IP Networks

[2] http://cr.yp.to/proto/utctai.html

[3] MySQL documentation: http://www.mysql.com/doc/en/