Захарова [и др.]. Юргинский технологический институт - Томск: Изд-во Томского политехнического университета, 2013. - 147 с.

3. Теория статистики [Текст] / Под ред. Р.А. Шмойловой - М.: "Финансы и статистика", 1996. - 416 с.

*Аспирант **Е.В. Ожогов** – blackjack41@mail.ru; - Юргинский технологический институт Томского политехнического университета, кафедра информационных систем, k_is@inbox.ru, http://uti.tpu.ru, (384-51)6-49-42.*

УДК: 519.857.6

DATA DIFFERENCING METHOD TO OPTIMIZE DATA STORING IN WEATHER MONITORING SYSTEM

H.M. Hussein A.G. Yakunin

This work aims to optimize the storage space for database that stores the measured data for weather parameter. The database is a part from a complete project that monitors the weather parameters for long period and stores a hung amount of data in the database. With time the size of the database grows up. After a long time, the database size will be very big, which needs a large storage space and takes a long time for processing.

Our method saves more than 78% of storage space, increases the data transfer rate and enhances the system resources usage.

To store the measured data, the proposed method stores the differences between the actual measured data. Storing the difference instead of the actual data saves the storage space by 78%.

Keywords: data compression, weather monitoring, weather prediction, temperature sensors, compression saving, data differencing.

Introduction

Weather predictions (forecasts) are made by collecting a huge amount of data for the current weather state. The prediction accuracy depends on the collected data amount. The collected data can be analyzed using one of the weather forecast methods [1-3].

As the amount of measured data grows up, the size of the database increases. After a long time, the size of database will be too large, which will absorb the system storage space. To overcome this problem, many compression algorithms were studied [4-10].

But these algorithms are general purpose algorithms and they are not the desired for our case.

Method description

The system database contains table that stores the measured values for weather parameters. It stores the measured values from sensors, the time of measuring, and the sensor id.

For simplicity, only one parameter has been observed and analyzed. This parameter is temperature.

The experimental project has 6 temperature sensors (S1, S2,…,S6). Two of them measure the temperature outside lab. And the others have been distributed in different places in the lab.

Every sampling time ”T_s” equals30s, these sensors measure temperature values in the range -55^o c and 125 ^{o} c and send it to the server. Users can explore these values with GUI website interface *(http://abc.altstu.ru).*

Snapshots for website interface are shown in fig. 1 and fig. 2 for outside and inside measured values.

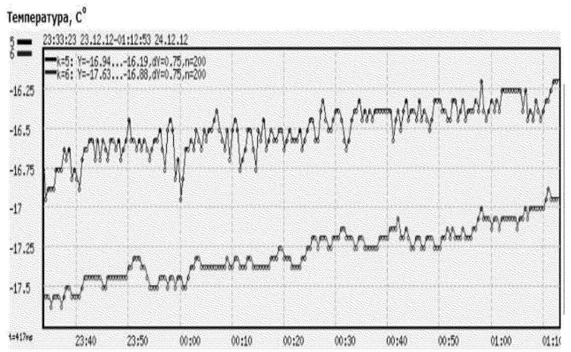

Fig.1 - Measured values for outside temperature.

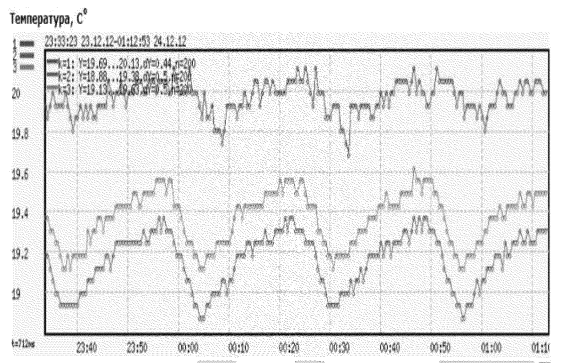

Fig.2 - Measured values for inside temperature.

It is noted that the temperature does not change suddenly unless there is a sudden change in some influencing factors.

As seen in the previous figures, the rates of temperature changes are very small. Also the differences between measure values from different sensors are small.

For analysis, two sensors, one outside (S1) and the other inside (S3), have been observed carefully throughout a full day. Their measured values were plotted in fig 3.

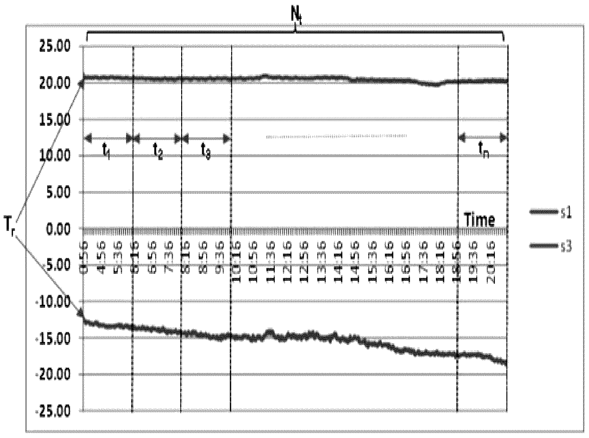

Fig 3 - Measured values from sensors S1 and S3.

To analysis the measured values, the average and standard deviation values were calculated as shown Table 1.

The calculations indicate that the standard deviation of the measured values is small. So, it's better to store the differences between successive temperature values to save the storage space.

Moreover, to save space, it is not necessary to store the time of measurements (knowable implicitly), where the measurements are made at regular time intervals.

Table 1: Analysis results for measured values.

	S1	S2
Max. value	-12.56	21.00
Min. value	-18.63	19.69
Average	-15.26	20.52
SD	1.42	0.25
dT_{max}	0.44	0.06
dT_{min}	-0.38	-0.13
$dT_{=0}$	23.1%	73.55%
$dT_{<0.1}$	83 %	100%
$dT_{average}$	0.00	0.00
dT_{SD}	0.11	0.03

Where: SD: standard deviation.

dT_{max}: The maximum value of the differences.
dT_{min}: The minimum value of the differences.
$dT_{=0}$: Difference values which are equals zeros.
$dT_{<0.1}$: Difference values less than 0.1.
$dT_{average}$: Average of difference values.
dT_{SD} : standard deviation of difference values.

It is noted that the statistics in Table 1 indicate that large percentage of the difference values is equal to zero. So, to save more storage space, those values are not stored. (And more space can be saved if the difference values less than acceptable tolerance value "dT_m" are not saved).

But if this modification will be applied, a problem will appear in retrieving data at a given time, since the time field will be removed from the database and data will be stored in an irregular way.

The following proposal will solve this problem:

- The time period in every day will be divided into fixed time intervals "N_t" (as shown in Fig. 3)
- The difference values of temperature at the intervals boundaries will be saved.
- When a difference value will be saved in the database, the number of non-stored values "N_m" before it will be also saved.
- To save the storage space, the difference value and the number of non-stored values will be saved in the same record.

If we assume that the size of database record will be "B" bits and the size needed to store the difference value "b_t", then the rest bits ($B-b_t$) will be used to store the number of non-stored values (N_m).

The size of B and b_t can be adjusted by the system administrator depending the system requirements.

Method procedure

The flowcharts shown in Fig.4 and Fig. 5 show how this method will work to store and retrieve the data.

Method analysis

To evaluate the compression method, the compression ratio and the compression saving will be calculated.

Compression Ratio (CR): It is the ratio between the size of compressed file and the size of the source file [4].

It can be calculated from the equation:

$$CR = \frac{\text{Size after compression}}{\text{Size before compression}}$$

The compression saving (CS) can be calculated using the following equation:

$$CS = \frac{\text{Size before compression} - \text{Size after compression}}{\text{Size before compression}}$$

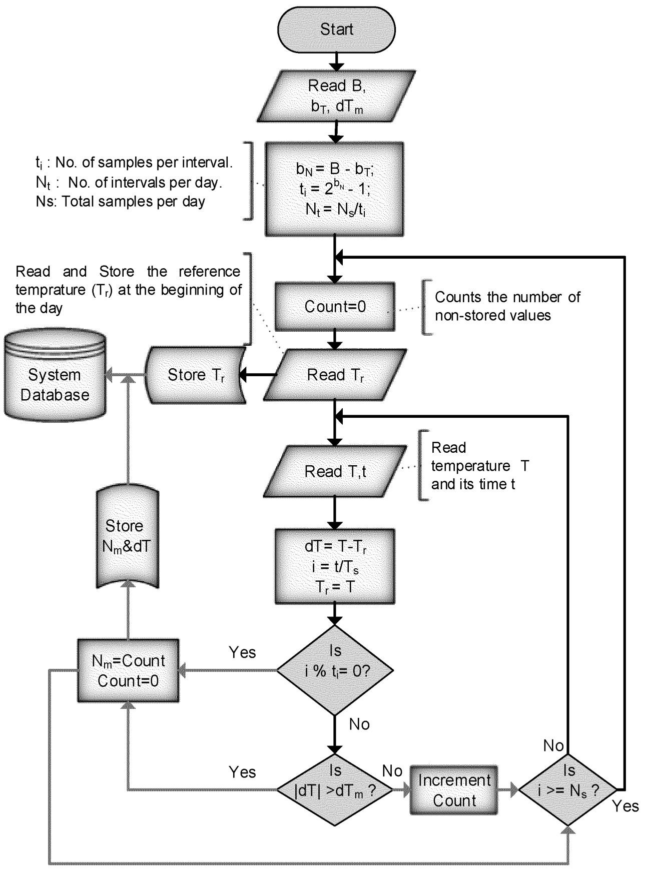
Start
Read B, b_T, dT_m
t_i : No. of samples per interval.
N_t : No. of intervals per day.
Ns: Total samples per day
b_N = B - b_T; $t_i = 2^{b_N} - 1$; $N_t = N_s/t_i$
Read and Store the reference temprature (T_r) at the beginning of the day
Count=0
Counts the number of non-stored values
System Database
Store T_r
Read T_r
Read T,t
Read temperature T and its time t
Store N_m&dT
dT= T-T_r; i = t/T_s; T_r = T
N_m=Count Count=0
Yes
Is i % t_i= 0?
No
Yes
Is |dT| >dT_m ?
No
Increment Count
No
Is i >= N_s ?
Yes

Fig. 4 - Procedure for storing data in the database

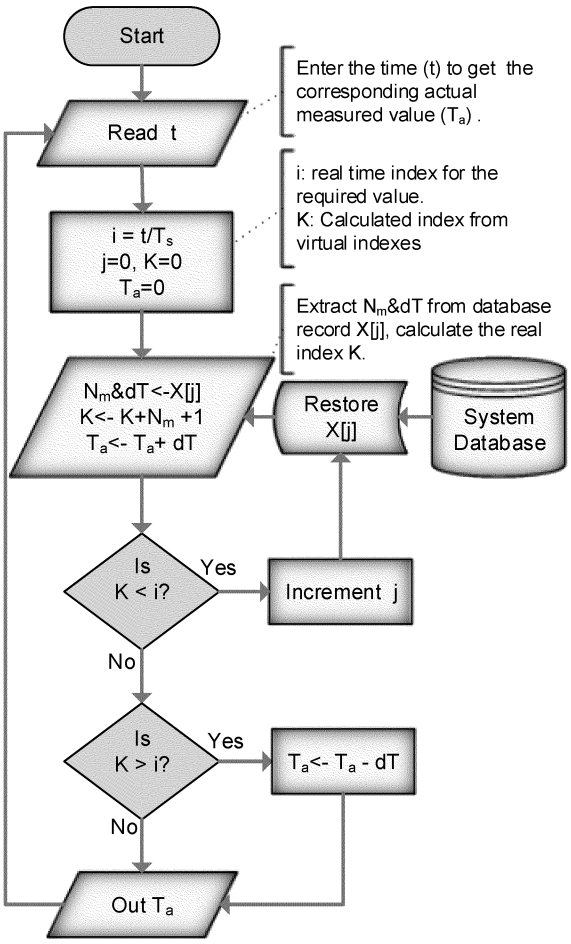
Start
Enter the time (t) to get the corresponding actual measured value (T_a) .
Read t
i: real time index for the required value.
K: Calculated index from virtual indexes
i = t/T_s; j=0, K=0; T_a=0
Extract N_m&dT from database record X[j], calculate the real index K.
N_m&dT<-X[j]; K<- K+N_m +1; T_a<- T_a+ dT
Restore X[j]
System Database
Is K < i?
Yes
Increment j
No
Is K > i?
Yes
T_a<- T_a - dT
No
Out T_a

***Fig. 5 -** Procedure for retrieving data from the database*

For each temperature value, two bytes were needed for regular method. But when the difference values are stored instead the actual values, only one byte will be needed, because the difference values vary in the range ±0.5°c (as shown in table 1).

Then, CR= $\frac{1\ byte}{2\ byte}$ = 50%.

Also CS = 50%.

Also, for lossless compression, the difference values equal zeros (dT_m=0) will not be stored. As shown in table 1, nearly 23% of outside sensors difference values are equal zeros, and more than 73% for inside difference values.

Then: $CS_{outside}$ = 23% *50% +50%=61.5%

CS_{inside} = 73% *50% +50%=86.5%

There are two outside sensors and four inside then:

$$CS_{average} = \frac{2*61.5\%+4*86.5\%}{6} = 78.17\%.$$

So, the investigated method will saves more than 78% of the storage space.

This saving ratio can be obtained using other compression methods. But this method has several features, including:

- It works with online database (not dumped data).
- It can be used at terminal computer which collects data from sensors, so the data traffic to the server will be faster and lighter.
- It is fast and easy.
- It doesn't absorb the system resources.
- The database data transfer will be faster.
- It may be used in data errors detection and correction.
- Also it may be used for detecting the hazards of sensors operations or turbulence cases if the difference value exceeds the acceptable tolerance.

Conclusion and future work

An optimized method for storing weather measured data in the database has been investigated and evaluated. The method saves more than 78% of the storage space.

It may be also used for data errors detection and correction. For that an algorithm may be investigated. The method has been applied for temperature measurements and it can be extended for other weather parameters with small modification.

REFERENCES

1. Glahn, H. The Use of Model Output Statistics (MOS) in Objective Weather Forecasting/ H. Glahn and D. Lowry// Journal of Applied Meteorology, vol. 11, Issue 8, Dec. 1972, pp. 1203-1211.
2. Gringorten, I. Methods of Objective Weather Forecasting/ I. Gringorten// Advances in Geophysics Volume 2, 1955, pp 57–92.
3. Leith, C. Objective Methods for Weather Prediction/ C. Leith// Annual Review of Fluid Mechanics, Jan. 1978, Vol. 10, Pages 107-128.
4. Saloman, D. Data Compression the Complete Reference/ D. Saloman//Springer, 3rd Edition (2004).
5. Zhen, Ch. Design and Realization of Data Compression in Real-Time Database/ Ch. Zhen and B. Ren// IEEE International Conference on Computational Intelligence and Software Engineering, China, 11 – 13 Dec. 2009 pp. 340-243.
6. Yang, H. On the performance of data compression algorithms based upon string matching/ H. Yang// IEEE Transactions on Information Theory, Jan 1998, Vol. 44, Issue 1, pp.47- 65.
7. Lin, M. A New Architecture of a Two-Stage Lossless Data Compression and Decompression Algorithm/ M. Lin and Y. Yi Chang//IEEE transactions on VLSI, Vol. 17, Issue 9, Sept. 2009, pp. 1297- 1303.
8. Ray, G. Data Compression in Databases/ G. Ray//Master's Thesis, Dept. of Computer Science and Automation, Indian Institute of Science, June 1995.
9. Tamrakar, A. A Compression Algorithm for Optimization of Storage Consumption of Non Oracle Database/A. Tamrakar and V. Nanda//A Compression Algorithm for Optimization of Storage Consumption of Non Oracle Database, July 2012, Vol. 1, Issue 5, pp. 39-43.
10. Всё о сжатии данных, изображений и видео [Electronic resource]/Access mode: http:// www. compression.ru.- Title from screen (last visit on 17-03-2013).

*Аспирант **Хуссейн М.Х.**, helphs@yahoo.com, Проф. **Якунин А.Г.**, тел. (8(3852) 29-07-86. e-mail: lis@agtu.secna.ru., Россия, 656038, г. Барнаул, пр-т Ленина, 46, Алтайский государственный технический университет им.И.И.Ползунова, Факультет информационных технологий, кафедра вычислительных систем и информационной безопасности,*

УДК: 519.687.1

ИНСТРУМЕНТАЛЬНАЯ ПОДДЕРЖКА ЭФФЕКТИВНОГО ИСПОЛЬЗОВАНИЯ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ГЕТЕРОГЕННЫХ СИСТЕМ

С.А. Гризан, А.И. Легалов

Рассматриваются подходы к оптимизации программ, предназначенных для выполнения вычислений на графических ускорителях. Представлены алгоритмы балансировки нагрузки, зависящие от критических параметров решаемой задачи. Приведены примеры использования разработанной библиотеки, обеспечивающей подстройку решаемой задачи под используемую конфигурацию вычислительных средств.

Ключевые слова: GPU, CUDA, автоподстройка, оптимизация.

Введение

В настоящее время наблюдается широкое использование суперкомпьютерных систем на основе неоднородной (гетерогенной) архитектуры. Основными вычислительными элементами в них являются специализированные графические ускорители (GPU), позволившие обойти кластерные системы как по пиковой, так и по реально достигнутой производительности. Вместе с тем, графические ускорители, являясь одной из перспективных вычислительных платформ, требуют для своего эффективного использования дополнительных затрат на анализ методов распараллеливания решаемой прикладной задачи и их программирование. Для сокращения этих затрат и создания эффективных параллельных программ необходимы методы автоматизации программирования, направленные на оптимизацию кода для ключевых вычислительных ядер, под которыми в прикладных задачах понимаются фрагменты кода, реализующие параллельные алгоритмы. Вычислительные ядра при этом могут быть автоматически подстроены под конкретное приложение как в контексте определенных входных и